

NNN	NNN	MMM	MMM	LLL
NNN	NNN	MMM	MMM	LLL
NNN	NNN	MMM	MMM	LLL
NNN	NNN	MMMMMM	MMMMMM	LLL
NNN	NNN	MMMMMM	MMMMMM	LLL
NNN	NNN	MMMMMM	MMMMMM	LLL
NNNNNN	NNN	MM	MM	LLL
NNNNNN	NNN	MM	MM	LLL
NNNNNN	NNN	MM	MM	LLL
NN NNN	NNN	MM	MM	LLL
NN NNN	NNN	MM	MM	LLL
NN NNN	NNN	MM	MM	LLL
NNN	NNNNNN	MM	MM	LLL
NNN	NNNNNN	MM	MM	LLL
NNN	NNNNNN	MM	MM	LLL
NNN	NNN	MM	MM	LLL
NNN	NNN	MM	MM	LLL
NNN	NNN	MM	MM	LLL
NNN	NNN	MM	LLLLLLLLLLLL	
NNN	NNN	MM	LLLLLLLLLLLL	
NNN	NNN	MM	LLLLLLLLLLLL	

FILEID**NMLLOGOPS

NN NN MM MM LL LL 000000 GGGGGGGGGG 000000 PPPPPPPP SSSSSSSS
NN NN MM MM LL LL 000000 GGGGGGGGGG 000000 PPPPPPPP SSSSSSSS
NN NN MMMM MMMM LL LL 00 00 GG 00 00 PP PP SS
NN NN MMMM MMMM LL LL 00 00 GG 00 00 PP PP SS
NNNN NN MM MM LL LL 00 00 GG 00 00 PP PP SS
NNNN NN MM MM LL LL 00 00 GG 00 00 PP PP SS
NN NN NN MM MM LL LL 00 00 GG 00 00 PPPPPPPP SSSSSS
NN NN NN MM MM LL LL 00 00 GG 00 00 PPPPPPPP SSSSSS
NN NNNN MM MM LL LL 00 00 GG GGGGGG 00 00 PP SS
NN NNNN MM MM LL LL 00 00 GG GGGGGG 00 00 PP SS
NN NN MM MM LL LL 00 00 GG GG 00 00 PP SS
NN NN MM MM LL LL 00 00 GG GG 00 00 PP SS
NN NN MM MM LLLLLLLL LLLLLLLL 000000 GGGGGG 000000 PP SSSSSSSS
NN NN MM MM LLLLLLLL LLLLLLLL 000000 GGGGGG 000000 PP SSSSSSSS

LL IIIII SSSSSSSS
LL IIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LLLLLLL LLLL LIII IIII SSSSSSSS SSSSSSSS

NM
VO

```
1 0001 0 %TITLE 'NML Logging data base operations module'
2 0002 0 MODULE NML$LOGOPS {
3 0003 0   LANGUAGE (BLISS32),
4 0004 0   ADDRESSING_MODE (EXTERNAL=LONG_RELATIVE),
5 0005 0   ADDRESSING_MODE (NONEXTERNAL=LONG_RELATIVE),
6 0006 0   IDENT = 'V04-000'
7 0007 0   )
8 0008 1 BEGIN
9 0009 1 ****
10 0010 1 *
11 0011 1 *
12 0012 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
13 0013 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
14 0014 1 * ALL RIGHTS RESERVED.
15 0015 1 *
16 0016 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
17 0017 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
18 0018 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
19 0019 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
20 0020 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
21 0021 1 * TRANSFERRED.
22 0022 1 *
23 0023 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
24 0024 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
25 0025 1 * CORPORATION.
26 0026 1 *
27 0027 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
28 0028 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
29 0029 1 *
30 0030 1 *
31 0031 1 ****
32 0032 1 *
33 0033 1 *
34 0034 1 ++
35 0035 1 | FACILITY: DECnet-VAX V2.0 Network Management Listener
36 0036 1 |
37 0037 1 | ABSTRACT:
38 0038 1 |
39 0039 1 | These routines handle all logging data base operations.
40 0040 1 |
41 0041 1 | ENVIRONMENT: VAX/VMS Operating System
42 0042 1 |
43 0043 1 | AUTHOR: Distributed Systems Software Engineering
44 0044 1 |
45 0045 1 | CREATION DATE: 26-JUN-1980
46 0046 1 |
47 0047 1 | MODIFIED BY:
48 0048 1 |   V03-002 MKP0002 Kathy Perko 23-Nov-1982
49 0049 1 |   Add module as a source for events.
50 0050 1 |
51 0051 1 |   V02-001 MKP0001 Kathy Perko 16-Nov-1981
52 0052 1 |   Add circuit entity as a logging source type.
53 0053 1 |
54 0054 1 | --
55 0055 1 | --
```

```
: 57      0056 1 %SBTTL 'Declarations'  
: 58      0057 1  
: 59      0058 1 !  
: 60      0059 1 ! TABLE OF CONTENTS:  
: 61      0060 1 !  
: 62      0061 1  
: 63      0062 1 FORWARD ROUTINE  
: 64      0063 1     NML$ADDFILTERS,  
: 65      0064 1     NML_MODFIL,  
: 66      0065 1     NML_MODCLS,  
: 67      0066 1     NML_MODKNO,  
: 68      0067 1     NML$GETSPCFILTERS,  
: 69      0068 1     NML$GETCOMFILTERS,  
: 70      0069 1     NML$GETGBLFILTERS,  
: 71      0070 1     NML$CLEANEVT      : NOVALUE,  
: 72      0071 1     NML$CLEANSRC      : NOVALUE,  
: 73      0072 1     NML$MATCHSRC,  
: 74      0073 1     NML$GETNXTSNK,  
: 75      0074 1     NML$GETNXTSRC,  
: 76      0075 1     NML$MATCHEVT,  
: 77      0076 1     NML$GETNXTEVT,  
: 78      0077 1     NML$BLDSRC      : NOVALUE,  
: 79      0078 1     NML$BLDEVT      : NOVALUE,  
: 80      0079 1     NML$ADDSRC,  
: 81      0080 1     NML$REPSRC,  
: 82      0081 1     NML$REMSRC      : NOVALUE,  
: 83      0082 1     NML$ADDEVT,  
: 84      0083 1     NML$MODEVT      : NOVALUE,  
: 85      0084 1     NML$REMEVT      : NOVALUE;  
: 86      0085 1  
: 87      0086 1 !  
: 88      0087 1 ! INCLUDE FILES:  
: 89      0088 1 !  
: 90      0089 1  
: 91      0090 1 LIBRARY 'LIB$:NMLLIB.L32';  
: 92      0091 1 LIBRARY 'SHRLIB$:NMALIBRY.L32';  
: 93      0092 1 LIBRARY 'SYSSLIBRARY:STARLET.L32';  
: 94      0093 1  
: 95      0094 1 !  
: 96      0095 1 ! OWN STORAGE:  
: 97      0096 1 !  
: 98      0097 1  
: 99      0098 1 OWN  
:100      0099 1     NML$T_EVTBUFFER : BBLOCK [EVT$K_LENGTH],  
:101      0100 1     NML$T_SRCBUFFER : BBLOCK [NML$K_RECVFLÉN];  
:102      0101 1 BIND  
:103      0102 1     NML$Q_EVTBFDESC = UPLIT (EVT$K_LENGTH, NML$T_EVTBUFFER) : DESCRIPTOR,  
:104      0103 1     NML$Q_SRCBFDESC = UPLIT (NML$K_RECVFLÉN, NML$T_SRCBUFFER) : DESCRIPTOR;  
:105      0104 1  
:106      0105 1 !  
:107      0106 1 ! EXTERNAL REFERENCES:  
:108      0107 1 !  
:109      0108 1  
:110      0109 1 $NML_EXTDEF:  
:111      0110 1  
:112      0111 1 EXTERNAL LITERAL  
:113      0112 1     NML$GK_EVENTS;
```

```
: 114      0113 1
: 115      0114 1 EXTERNAL
: 116      0115 1   NML$AB_EVENTS : BBLOCKVECTOR [0, ETB$K_ENTRYLEN];
: 117      0116 1
: 118      0117 1 EXTERNAL ROUTINE
: 119      0118 1   NML$ERROR_2;
: 120      0119 1
```

```
122      0120 1 %SBTTL 'NML$ADDFILTERS Add event filters for sink node'
123      0121 1 GLOBAL ROUTINE NML$ADDFILTERS
124          (FCT, BUFDSC, SNK, SRC, ENTDSC, CLASS, MSKLEN, MSKPTR, RESDSC) =
125
126      0124 1 ++
127      0125 1 FUNCTIONAL DESCRIPTION:
128          0126 1
129              This routine adds event filters to the data base entry for a sink
130              node.
131          0127 1
132          0130 1 FORMAL PARAMETERS:
133          0131 1
134          0132 1     FCT           Function code. (0=CLEAR/PURGE, 1=SET/DEFINE)
135          0133 1     BUFDSC        Descriptor of buffer to contain modified data base
136          0134 1             entry.
137          0135 1     SNK            Logging sink type code.
138          0136 1     SRC            Event source type code.
139          0137 1     ENTDSC         Event source id string descriptor.
140          0138 1     CLASS          Event class code.
141          0139 1     MSKLEN         Length of filter mask.
142          0140 1     MSKPTR         Address of filter mask.
143          0141 1     RESDSC         Descriptor of data in buffer.
144
145          0142 1
146          0143 1     IMPLICIT INPUTS:
147          0144 1
148          0145 1     NML$GB_EVTMSKTYP
149          0146 1
150          0147 1     IMPLICIT OUTPUTS:
151          0148 1
152          0149 1     NONE
153
154          0150 1     ROUTINE VALUE:
155          0151 1     COMPLETION CODES:
156
157          0152 1
158          0153 1
159          0154 1     TRUE is returned if operation is successful. Otherwise, FALSE
160          0155 1     is returned.
161
162          0156 1
163          0157 1     SIDE EFFECTS:
164
165          0158 1
166          0159 1
167
168          0160 1
169          0161 1     --
170
171          0162 1
172
173          0163 2     BEGIN
174
175          0164 2
176          0165 2     MAP
177
178          0166 2     BUFDSC : REF DESCRIPTOR,
179          0167 2     ENTDSC : REF DESCRIPTOR,
180          0168 2     RESDSC : REF DESCRIPTOR;
181
182          0169 2     LOCAL
183
184          0170 2     SRCPTR : REF BBLOCK,           ! Pointer to source block
185          0171 2     STATUS;                  ! Routine status code
186
187          0172 2     STATUS = TRUE;            ! Initialize return status
188
189          0173 2
190
191          0174 2     ! Get the source block.
```

```
179      0177 2 !
180      0178 2 ! IF NML$MATCHSRC (.RESDSC, .SNK, .SRC, .ENTDSC, SRCPTR)
181      0179 2 THEN
182          0180 3 BEGIN
183          0181 3
184          0182 3     CH$MOVE (.SRCPTR [SRC$W_LENGTH],
185          0183 3             .SRCPTR,
186          0184 3             NML$T_SRCBUFFER);
187          0185 3     NML$REMSRC (.RESDSC, .SRCPTR);
188          0186 3     SRCPTR = NML$T_SRCBUFFER;
189          0187 3
190          0188 3 END
191          0189 2 ELSE
192              0190 3 BEGIN
193              0191 3
194              0192 3     NML$BLDSRC (NML$Q_SRCBFDS, .SNK, .SRC, .ENTDSC);
195              0193 3     SRCPTR = .NML$Q_SRCBFDS [DSC$A_POINTER];
196              0194 3
197              0195 2 END;
198              0196 2 !
199              0197 2 ! Add the events to the source block.
200              0198 2 !
201              0199 2 SELECTONEU .NML$GB_EVTMSKTYP OF
202                  0200 2 SET
203                  0201 2
204                  0202 2 [2]: ! All events in class
205                  0203 2
206                  0204 2     NML_MODCLS (.FCT, NML$Q_SRCBFDS, .SRCPTR, .CLASS, .SRC);
207                  0205 2
208                  0206 2 [3]: ! Known events
209                  0207 2
210                  0208 2     NML_MODKNO (.FCT, NML$Q_SRCBFDS, .SRCPTR, .SRC);
211                  0209 2
212                  0210 2 [OTHERWISE]: ! Add specified events to class
213                  0211 2
214                  0212 2     NML_MODFIL (.FCT,
215                  0213 2         FALSE,
216                  0214 2         NML$Q_SRCBFDS,
217                  0215 2         .SRCPTR,
218                  0216 2         .CLASS,
219                  0217 2         .MSKLEN,
220                  0218 2         .MSKPTR);
221                  0219 2
222                  0220 2 TES;
223                  0221 2
224                  0222 2 ! Add the source block to the data base entry.
225                  0223 2
226                  0224 2 IF NOT NML$ADDSRC (.BUFDSC, .RESDSC, .SRCPTR)
227                  0225 2 THEN
228                      0226 2     STATUS = FALSE;
229                      0227 2
230                      0228 2 ! Clean up the sink node filters.
231                      0229 2
232                      0230 2     NML$CLEANEVT (.SNK, .RESDSC);
233                      0231 2     NML$CLEANSRC (.BUFDSC, .SNK, .RESDSC);
234                      0232 2
235                      0233 2 RETURN .STATUS
```

: 236 0234 2
: 237 0235 1 END:

! End of NML\$ADDFILTERS

```
.TITLE NML$LOGOPS NML Logging data base operations module
.IDENT \V04-000\
.PSECT SPLITS,NOWRT,NOEXE,2

00000014, 00000 P.AAA: .LONG 20
00000000, 00004 .ADDRESS NMLST_EVTBUFFER
00000400, 00008 P.AAB: .LONG 1024
00000000, 0000C .ADDRESS NMLST_SRCBUFFER

.PSECT $OWNS,NOEXE,2

00000 NMLST_EVTBUFFER:
.BLKB 20
00014 NMLST_SRCBUFFER:
.BLKB 1024

NML$Q_EVTBFDESC= P.AAA
NML$Q_SRCBFDESC= P.AAB
.EXTRN NML$GB_EVTSRCTYP
.EXTRN NML$GQ_EVTSRCDSC
.EXTRN NML$GW_EVTCLASS
.EXTRN NML$GB_EVTMSKTYP
.EXTRN NML$GQ_EVTMSKDSC
.EXTRN NML$GW_EVTSNKADR
.EXTRN NML$GW_ACP_CHAN
.EXTRN NML$GL_LOGMASK, NML$GQ_ENTSTRDSC
.EXTRN NML$AB_QIOBUFFER
.EXTRN NML$GQ_QIOBFDSC
.EXTRN NML$AB_EXEBUFFER
.EXTRN NML$GL_EXEDATPTR
.EXTRN NML$GQ_EXEDATDSC
.EXTRN NML$GQ_EXEBFDSC
.EXTRN NML$AB_RCVBUFFER
.EXTRN NML$GQ_RCVbfdsc
.EXTRN NML$AB_SNDBUFFER
.EXTRN NML$GQ_SNDBFDSC
.EXTRN NML$GL_RCVDATLEN
.EXTRN NML$AB_CPTABLE, NML$AB_MSGBLOCK
.EXTRN NML$AB_ENTITY_ID
.EXTRN NML$AB_QUALIFIER_ID
.EXTRN NML$AB_ENTITYDATA
.EXTRN NML$AB_NML_NMV, NML$AB_PRMSEM
.EXTRN NML$AB_RECBUF, NML$AL_ENTINFTAB
.EXTRN NML$AL_PERMINFTAB
.EXTRN NML$AW_PRMDES, NML$GB_CMD_VER
.EXTRN NML$GB_ENTITY_CODE
.EXTRN NML$GB_ENTITY_FORMAT
.EXTRN NML$GL_QUALIFIER_PST
.EXTRN NML$GB_QUALIFIER_FORMAT
.EXTRN NML$GB_FUNCTION
.EXTRN NML$GB_INFO, NML$GB_OPTIONS
```

			.EXTRN NML\$GL_PRMCODE, NML\$GL_PRS_FLGS	
			.EXTRN NML\$GL_NML_ENTITY	
			.EXTRN NML\$GQ_NETNAMDSC	
			.EXTRN NML\$GQ_RECBLFDSC	
			.EXTRN NML\$GW_PRMDESCNT	
			.EXTRN NML\$GK_EVENTS, NML\$AB_EVENTS	
			.EXTRN NML\$ERROR_2	
			.PSECT \$CODE\$, NOWRT, 2	
			.ENTRY NML\$ADDFILTERS, Save R2,R3,R4,R5,R6,R7,R8,- ; 0121	
			R9	
			MOVAB NML\$T_SRCBUFFER, R9	
			MOVAB NML\$Q_SRCBLFDSC, R8	
			SUBL2 #4, SP	
			MOVL #1, STATUS	
			PUSHL SP	
			MOVQ SRC, -(SP)	
			PUSHL SNK	
			MOVL RESDSC, R6	
			PUSHL R6	
			CALLS #5, NML\$MATCHSRC	
			BLBC R0, 1\$	
			MOVC3 @SRCPTR, @SRCPTR, NML\$T_SRCBUFFER	
69	00	BE	00 03FC 00000	0182
			PUSHL SRCPTR	0185
			R6	
			CALLS #2, NML\$REMSRC	
			NML\$T_SRCBUFFER, SRCPTR	
			BRB 2\$	
			MOVQ SRC, -(SP)	
			PUSHL SNK	
			PUSHL R8	
			CALLS #4, NML\$BLDSRC	
			MOVL NML\$Q_SRCBLFDSC+4, SRCPTR	
			MOVZBL NML\$GB_EVTMSKTYPE, R0	
			CMPB R0, #2	
			BNEQ 3\$	
			PUSHL SRC	
			PUSHL CLASS	
			PUSHL SRCPTR	
			PUSHL R8	
			PUSHL FCT	
			CALLS #5, NML_MODCLS	
			BRB 5\$	
			CMPB R0, #3	
			BNEQ 4\$	
			PUSHL SRC	
			PUSHL SRCPTR	
			PUSHL R8	
			PUSHL FCT	
			CALLS #4, NML_MODKNO	
			BRB 5\$	
			MOVQ MSKLEN, -(SP)	
			PUSHL CLASS	
			PUSHL SRCPTR	
			PUSHL R8	
			CLRL -(SP)	
			0217	
			0216	
			0215	
			0212	

NML\$LOGOPS
V04-000

NML Logging data base operations module
NMLSADDFILTERS Add event filters for sink node

M 8

16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 8
(3)

00000000V	EF	04	AC DD 000A3	PUSHL	FCT	
		07	FB 000A6	CALLS	#7, NML_MODFIL	
		6E	DD 000AD	PUSHL	SRCPTR	0224
		56	DD 000AF	PUSHL	R6	
00000000V	EF	08	AC DD 000B1	PUSHL	BUFDSC	
	02	03	FB 000B4	CALLS	#3, NML\$ADDSRC	
		50	E8 000BB	BLBS	R0, 6\$	0226
		57	D4 000BE	CLRL	STATUS	
00000000V	EF	0C	AC DD 000C0	PUSHL	R6	0230
		02	FB 000C5	CALLS	#2, NML\$CLEANEVT	
		56	DD 000CC	PUSHL	R6	0231
00000000V	EF	08	AC 7D 000CE	MOVQ	BUFDSC, -(SP)	
	7E	03	FB 000D2	CALLS	#3, NML\$CLEANSRC	
	50	57	DD 000D9	MOVL	STATUS, R0	0233
		04	000DC	RET		0235

; Routine Size: 221 bytes, Routine Base: \$CODE\$ + 0000

```
: 239      0236 1 %SBTTL 'NML_MODFIL  Modify event filters'  
: 240      0237 1 ROUTINE NML_MODFIL (FCT, ZER, BUFDSC, SRCPTR, CLASS, MSKLEN, MSKPTR) =  
: 241      0238 1  
: 242      0239 1 ++  
: 243      0240 1 FUNCTIONAL DESCRIPTION:  
: 244      0241 1  
: 245      0242 1 This routine adds event filters to the data base entry for a sink  
: 246      0243 1 node.  
: 247      0244 1  
: 248      0245 1 FORMAL PARAMETERS:  
: 249      0246 1  
: 250      0247 1 FCT          Function code. (0=CLEAR/PURGE, 1=SET/DEFINE).  
: 251      0248 1 ZER           Zero mask flag. (TRUE=yes, FALSE=no).  
: 252      0249 1 BUFDSC        Descriptor of buffer to contain modified data base  
: 253      0250 1 entry.  
: 254      0251 1 SRCPTR        Pointer to source block in buffer.  
: 255      0252 1 CLASS          Event class code.  
: 256      0253 1 MSKLEN         Length of filter mask.  
: 257      0254 1 MSKPTR         Address of filter mask.  
: 258      0255 1  
: 259      0256 1 IMPLICIT INPUTS:  
: 260      0257 1  
: 261      0258 1     NONE  
: 262      0259 1  
: 263      0260 1 IMPLICIT OUTPUTS:  
: 264      0261 1  
: 265      0262 1     NONE  
: 266      0263 1  
: 267      0264 1 ROUTINE VALUE:  
: 268      0265 1 COMPLETION CODES:  
: 269      0266 1  
: 270      0267 1     TRUE is returned if operation is successful. Otherwise, FALSE  
: 271      0268 1     is returned.  
: 272      0269 1  
: 273      0270 1 SIDE EFFECTS:  
: 274      0271 1  
: 275      0272 1     NONE  
: 276      0273 1  
: 277      0274 1 --  
: 278      0275 1  
: 279      0276 2 BEGIN  
: 280      0277 2  
: 281      0278 2 MAP  
: 282      0279 2     BUFDSC : REF DESCRIPTOR,  
: 283      0280 2     SRCPTR : REF BBLOCK;  
: 284      0281 2  
: 285      0282 2 LOCAL  
: 286      0283 2     EVTPTR,           ! Pointer to event block  
: 287      0284 2     STATUS;            ! Routine status code  
: 288      0285 2  
: 289      0286 2     STATUS = TRUE;       ! Initialize return status  
: 290      0287 2  
: 291      0288 2 Get the event block.  
: 292      0289 2  
: 293      0290 2 IF NMLSMATCHEVT (.SRCPTR,  
: 294      0291 2     .CLASS,  
: 295      0292 2     .EVTPTR)
```

```

: 296    0293 2      THEN
: 297    0294 3      BEGIN
: 298    0295 3
: 299    0296 3      NML$MODEVT (.FCT, .ZER, .EVTPTR, .MSKLEN, .MSKPTR);
: 300    0297 3
: 301    0298 3      END
: 302    0299 2      ELSE
: 303    0300 3      BEGIN
: 304    0301 3
: 305    0302 3      NML$BLDEVT (.FCT, .CLASS, .MSKLEN, .MSKPTR, NML$T_EVTBUFFER);
: 306    0303 3      EVT PTR = NML$T_EVTBUFFER;
: 307    0304 3
: 308    0305 3      | Add the event block to the source block.
: 309    0306 3
: 310    0307 3      IF NOT NML$ADDEVT (.BUFDSC, .SRCPTR, .EVTPTR)
: 311    0308 3      THEN
: 312    0309 3      STATUS = FALSE;
: 313    0310 3
: 314    0311 2      END;
: 315    0312 2
: 316    0313 2      RETURN .STATUS
: 317    0314 2
: 318    0315 1      END:           ! End of NML_MODFIL

```

000C 00000 NML_MODFIL:

					.WORD	Save R2,R3	0237
53	00000000'	EF	9E 00002	MOVAB	NML\$T_EVTBUFFER, R3		
5E		04	C2 00009	SUBL2	#4, SP	0286	
52		01	D0 0000C	MOVL	#1, STATUS	0290	
		5E	DD 0000F	PUSHL	SP		
00000000V	7E	10	AC 7D 00011	MOVQ	SRCPTR, -(SP)		
	EF	03	FB 00015	CALLS	#3, NML\$MATCHEV		
	14	50	E9 0001C	BLBC	RO, 1\$	0296	
	7E	18	AC 7D 0001F	MOVQ	MSKLEN, -(SP)		
		08	AE DD 00023	PUSHL	EVT PTR		
00000000V	7E	04	AC 7D 00026	MOVQ	FCT, -(SP)		
	EF	05	FB 0002A	CALLS	#5, NML\$MODEVT		
		28	11 00031	BRB	2\$	0290	
		53	DD 00033	1\$: PUSHL	R3	0302	
	7E	18	AC 7D 00035	MOVQ	MSKLEN, -(SP)		
		14	AC DD 00039	PUSHL	CLASS		
00000000V	04	AC	DD 0003C	PUSHL	FCT		
	EF	05	FB 0003F	CALLS	#5, NML\$BLDEVT		
	6E	63	9E 00046	MOVAB	NML\$T_EVTBUFFER, EVT PTR	0303	
		6E	DD 00049	PUSHL	EVT PTR	0307	
00000000V	7E	0C	AC 7D 0004B	MOVQ	BUFDSC, -(SP)		
	EF	03	FB 0004F	CALLS	#3, NML\$ADDEVT		
	02	50	E8 00056	BLBS	RO, 2\$	0309	
		52	D4 00059	CLRL	STATUS	0313	
	50	52	DO 0005B	2\$: MOVL	STATUS, RO		
			04 0005E	RET		0315	

; Routine Size: 95 bytes. Routine Base: \$CODE\$ + 000D

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML_MODFIL Modify event filters

C 9
16-Sep-1984 00:19:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:50:11 [NML.SRC]NMLLOGOPS.B32;1

Page 11
(4)

NML
V04

```
0316 1 %SBTTL 'NML_MODCLS  Modify class filters'  
0317 1 ROUTINE NML_MODCLS (FCT, BUFDSC, SRCPTR, CLASS, SRC) =  
0318 1  
0319 1 ++  
0320 1 | FUNCTIONAL DESCRIPTION:  
0321 1 |  
0322 1 | This routine adds event filters to the data base entry for a sink  
0323 1 | node.  
0324 1 |  
0325 1 | FORMAL PARAMETERS:  
0326 1 |  
0327 1 | FCT Function code. (0=CLEAR/PURGE, 1=SET/DEFINE)  
0328 1 | BUFDSC Descriptor of buffer to contain modified data base  
0329 1 | entry.  
0330 1 | SRCPTR Pointer to source block in buffer.  
0331 1 | CLASS Event class code.  
0332 1 | SRC Source type code.  
0333 1 |  
0334 1 | IMPLICIT INPUTS:  
0335 1 |  
0336 1 | NONE  
0337 1 |  
0338 1 | IMPLICIT OUTPUTS:  
0339 1 |  
0340 1 | NONE  
0341 1 |  
0342 1 | ROUTINE VALUE:  
0343 1 | COMPLETION CODES:  
0344 1 |  
0345 1 | TRUE is returned if operation is successful. Otherwise, FALSE  
0346 1 | is returned.  
0347 1 |  
0348 1 | SIDE EFFECTS:  
0349 1 |  
0350 1 | NONE  
0351 1 |  
0352 1 |  
0353 1 |  
0354 2 | --  
0355 2 | BEGIN  
0356 2 |  
0357 2 | MAP  
0358 2 | BUFDSC : REF DESCRIPTOR,  
0359 2 | SRCPTR : REF BBLOCK,  
0360 2 | CLASS : WORD;  
0361 2 |  
0362 2 | LOCAL  
0363 2 | MSK, ! Address of filter mask  
0364 2 | STATUS; ! Routine status code  
0365 2 | MSK = UPLIT (-1, 0);  
0366 2 |  
0367 2 | IF .FCT  
0368 2 | THEN  
0369 2 |  
0370 2 | INCR I FROM 0 TO NML$GK_EVENTS - 1 DO  
0371 2 | BEGIN  
0372 3 |  
0373 3 |  
0374 3 |  
0375 3 |  
0376 3 |
```

```

377      0373 3      IF .NML$AB_EVENTS [.I, ETBSW_CLASS] EQLU .CLASS
378      0374 3      THEN
379      0375 4      BEGIN
380      0376 4      SELECTONEU .SRC OF
381      0377 4      SET
382      0378 4
383      0379 4
384      0380 4      [NMASC_ENT_NOD]: ! Node
385      0381 4      MSR = .NML$AB_EVENTS [.I, ETBSA_NODE];
386      0382 4
387      0383 4      [NMASC_ENT_CIR]: ! Circuit
388      0384 4      MSR = .NML$AB_EVENTS [.I, ETBSA_CIRCUIT];
389      0385 4
390      0386 4      [NMASC_ENT_LIN]: ! Line
391      0387 4      MSR = .NML$AB_EVENTS [.I, ETBSA_LINE];
392      0388 4
393      0389 4      [NMASC_ENT_MOD]: ! Module
394      0390 4      MSR = .NML$AB_EVENTS [.I, ETBSA_MODULE];
395      0391 4
396      0392 4      [OTHERWISE]: ! Must be global
397      0393 4      MSR = .NML$AB_EVENTS [.I, ETBSA_GLOBAL];
398      0394 4
399      0395 4      TES;
400      0396 4
401      0397 4      EXITLOOP;
402      0398 4
403      0399 3      END;
404      0400 2
405      0401 2
406      0402 2      STATUS = NML_MODFIL (.FCT,
407                      TRUE,
408                      .BUFDESC,
409                      .SRCPTR,
410                      .CLASS,
411                      EVT$S_LOGMSK,
412                      .MSK);
413      0408 2
414      0410 2      RETURN .STATUS
415      0411 2
416      0412 1      END;                                ! End of NML_MODCLS

```

.PSECT \$PLIT\$,NOWRT,NOEXE,2

00000000 FFFFFFFF 00010 P.AAC: .LONG -1, 0

.PSECT \$CODE\$,NOWRT,2

001C 00000 NML_MODCLS:

54 00000000G	EF 9E 00002	.WORD	Save R2,R3,R4	: 0317
53 00000000	EF 9E 00009	MOVAB	NML\$AB_EVENTS, R4	: 0365
50 04	AC E9 00010	MOVAB	P.AAC_MSK	: 0367
50	01 CE 00014	BLBC	FCT, 8\$: 0373
		MNEGL	#1, I	

		43	11	00017	BRB	7\$				
		16	C5	00019	MULLS	#22, I, R1				
51	50	6441	9F	0001D	PUSHAB	NML\$AB_EVENTS[R1]				
		9E	B1	00020	CMPW	@(SP)+, CLASS				
		36	12	00024	BNEQ	7\$				
		14	AC	00026	MOVL	SRC, R2	0377			
		06	12	0002A	BNEQ	2\$	0380			
		06 A441	9F	0002C	PUSHAB	NML\$AB_EVENTS+6[R1]	0381			
		25	11	00030	BRB	6\$				
		52	D1	00032	CMPL	R2, #3	0383			
		06	12	00035	BNEQ	3\$				
		0A A441	9F	00037	PUSHAB	NML\$AB_EVENTS+10[R1]	0384			
		1A	11	0003B	BRB	6\$				
		52	D1	0003D	CMPL	R2, #1	0386			
		06	12	00040	BNEQ	4\$				
		0E A441	9F	00042	PUSHAB	NML\$AB_EVENTS+14[R1]	0387			
		0F	11	00046	BRB	6\$				
		04	52	00048	CMPL	R2, #4	0389			
		06	12	0004B	BNEQ	5\$				
		12 A441	9F	0004D	PUSHAB	NML\$AB_EVENTS+18[R1]	0390			
		04	11	00051	BRB	6\$				
		02 A441	9F	00053	5\$: PUSHAB	NML\$AB_EVENTS+2[R1]	0393			
		9E	D0	00057	6\$: MOVL	@(SP)+, MSK				
		08	11	0005A	BRB	8\$	0375			
B5	50 00000000G	8F	F3	0005C	7\$: AOBLEQ	#NML\$GK_EVENTS-1, I, 1\$	0370			
		53	DD	00064	8\$: PUSHL	MSK	0408			
		08	DD	00066	PUSHL	#8	0402			
		7E	10	AC	3C	00068	MOVZWL CLASS, -(SP)	0406		
		7E	08	AC	7D	0006C	MOVQ BUFDSC, -(SP)	0404		
					01	DD	PUSHL #1	0402		
					04	AC	DD	00072	PUSHL FCT	
	FF27	CF	07	FB	00075	CALLS	#7, NML_MODFIL			
					04	0007A	RET	0412		

; Routine Size: 123 bytes. Routine Base: \$CODE\$ + 013C

```
418      0413 1 %SBTTL 'NML_MODKNO  Modify known filters'  
419      0414 1 ROUTINE NML_MODKNO (FCT, BUFDSC, SRCPTR, SRC) =  
420      0415 1  
421      0416 1 ++  
422      0417 1 FUNCTIONAL DESCRIPTION:  
423      0418 1 This routine adds event filters to the data base entry for a sink  
424      0419 1 node.  
425      0420 1  
426      0421 1  
427      0422 1 FORMAL PARAMETERS:  
428      0423 1  
429      0424 1 FCT          Function code. (0=CLEAR/PURGE, 1=SET/DEFINE)  
430      0425 1 BUFDSC       Descriptor of buffer to contain modified data base  
431      0426 1 entry.  
432      0427 1 SRCPTR       Pointer to source block in buffer.  
433      0428 1 SRC          Source type code.  
434      0429 1  
435      0430 1 IMPLICIT INPUTS:  
436      0431 1     NONE  
437      0432 1  
438      0433 1 IMPLICIT OUTPUTS:  
439      0434 1     NONE  
440      0435 1  
441      0436 1  
442      0437 1  
443      0438 1 ROUTINE VALUE:  
444      0439 1 COMPLETION CODES:  
445      0440 1  
446      0441 1     TRUE is returned if operation is successful. Otherwise, FALSE  
447      0442 1 is returned.  
448      0443 1  
449      0444 1 SIDE EFFECTS:  
450      0445 1  
451      0446 1     NONE  
452      0447 1  
453      0448 1 --  
454      0449 1  
455      0450 2 BEGIN  
456      0451 2  
457      0452 2 MAP  
458      0453 2     BUFDSC : REF DESCRIPTOR,  
459      0454 2     SRCPTR : REF BBLOCK;  
460      0455 2  
461      0456 2 LOCAL  
462      0457 2     CLASS : WORD,  
463      0458 2     EVTPTR : REF BBLOCK,  
464      0459 2     MSK,  
465      0460 2     STATUS;           ! Routine status code  
466      0461 2  
467      0462 2 STATUS = FALSE;  
468      0463 2  
469      0464 2 INCR I FROM 0 TO NML$GK_EVENTS - 1 DO  
470      0465 3     BEGIN  
471      0466 3  
472      0467 3     CLASS = .NMLSAB_EVENTS [.I, ETBSW_CLASS];  
473      0468 3  
474      0469 3     SELECTONEU .SRC OF
```

```
; 475      0470 3      SET
; 476      0471 3
; 477      0472 3      [ENMASC_ENT_NOD]: ! Node
; 478      0473 3      MSR = .NML$AB_EVENTS [.I, ETBSA_NODE];
; 479      0474 3
; 480      0475 3      [ENMASC_ENT_CIR]: ! Circuit
; 481      0476 3      MSR = .NML$AB_EVENTS [.I, ETBSA_CIRCUIT];
; 482      0477 3
; 483      0478 3      [ENMASC_ENT_LIN]: ! Line
; 484      0479 3      MSR = .NML$AB_EVENTS [.I, ETBSA_LINE];
; 485      0480 3
; 486      0481 3      [ENMASC_ENT_MOD]: ! Line
; 487      0482 3      MSR = .NML$AB_EVENTS [.I, ETBSA_MODULE];
; 488      0483 3
; 489      0484 3      [OTHERWISE]: ! Must be global
; 490      0485 3      MSK = .NML$AB_EVENTS [.I, ETBSA_GLOBAL];
; 491      0486 3
; 492      0487 3      TES:
; 493      0488 3
; 494      0489 3      STATUS = NML_MODFIL (.FCT,
; 495          TRUE,
; 496          .BUFDSC,
; 497          .SRCPTR,
; 498          .CLASS,
; 499          EVT$S_LOGMSK,
; 500          .MSK);
; 501      0496 3      IF NOT .STATUS
; 502      0497 3      THEN
; 503      0498 3          EXITLOOP;
; 504      0499 3
; 505      0500 2      END;
; 506      0501 2
; 507      0502 2      If the function is clear and everything is alright up to this point then
; 508      0503 2      go through all event classes that are present in the source block and clear
; 509      0504 2      out all the filters. This covers the case where filters are present for
; 510      0505 2      an unknown class.
; 511      0506 2
; 512      0507 2      IF .STATUS
; 513      0508 2      AND NOT .FCT
; 514      0509 2      THEN
; 515      0510 3      BEGIN
; 516      0511 3
; 517      0512 3      EVTPTR = 0;
; 518      0513 3      WHILE NML$GETNXTEVT (.SRCPTR, EVTPTR) DO
; 519      0514 4      BEGIN
; 520      0515 4
; 521      0516 4      CLASS = .EVTPTR [EVT$W_CLASS];
; 522      0517 4      NML$MODEVT (.FCT, FALSE, .EVTPTR, EVT$S_LOGMSK, UPLIT (-1, -1));
; 523      0518 4
; 524      0519 3      END;
; 525      0520 3
; 526      0521 2
; 527      0522 2
; 528      0523 2      RETURN .STATUS
; 529      0524 2
; 530      0525 1      END;                                ! End of NML_MODKNO
```


NML\$LOGOPS
V04-000

NML Logging data base operations module
NML_MODRNO Modify known filters

J 9
16-Sep-1984 00:19:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:50:11 [NML.SRC]NMLLOGOPS.B32;1

Page 18
(6)

	08	AE	DD	00095	PUSHL	EVTPTR
		7E	D4	00098	CLRL	-(SP)
	04	AC	DD	0009A	PUSHL	FCT
00000000V	EF		05	FB	CALLS	#5, NML\$MODEVT
			D4	11	BRB	8\$
	50		54	00 000A6 9\$:	MOVL	STATUS, R0
			04	000A9	RET	

: 0513
: 0523
: 0525

; Routine Size: 170 bytes, Routine Base: \$CODE\$ + 01B7

```
: 532      0526 1 %SBTTL 'NML$GETSPCFILTERS Get event filters'
: 533      0527 1 GLOBAL ROUTINE NML$GETSPCFILTERS
: 534          (DATDSC, SNK, SRC, ENTDSC, CLASS, MSKPTR, RESLEN) =
: 535
: 536      0530 1 ++
: 537      0531 1 FUNCTIONAL DESCRIPTION:
: 538      0532 1 This routine gets event filters for the specified source and class.
: 539      0533 1
: 540      0534 1 FORMAL PARAMETERS:
: 541      0535 1
: 542      0536 1
: 543      0537 1 DATDSC      Descriptor of current data base entry.
: 544      0538 1 SNK          Logging sink type code.
: 545      0539 1 SRC          Event source type code.
: 546      0540 1 ENTDSC     Event source id string descriptor.
: 547      0541 1 CLASS        Event class code.
: 548      0542 1 MSKPTR      Address of filter mask quadword.
: 549      0543 1 RESLEN      Address of longword to contain byte count of
: 550      0544 1             resulting mask.
: 551      0545 1
: 552      0546 1 IMPLICIT INPUTS:
: 553      0547 1
: 554      0548 1
: 555      0549 1
: 556      0550 1 IMPLICIT OUTPUTS:
: 557      0551 1
: 558      0552 1
: 559      0553 1
: 560      0554 1 ROUTINE VALUE:
: 561      0555 1 COMPLETION CODES:
: 562      0556 1
: 563      0557 1 TRUE is returned if operation is successful. Otherwise, FALSE
: 564      0558 1 is returned.
: 565      0559 1
: 566      0560 1 SIDE EFFECTS:
: 567      0561 1
: 568      0562 1
: 569      0563 1
: 570      0564 1 --
: 571      0565 1
: 572      0566 2 BEGIN
: 573      0567 2
: 574      0568 2 MAP
: 575      0569 2 DATDSC : REF DESCRIPTOR,
: 576      0570 2 ENTDSC : REF DESCRIPTOR,
: 577      0571 2 MSKPTR : REF BITVECTOR;
: 578      0572 2
: 579      0573 2 LOCAL
: 580      0574 2 EVTPTR : REF BBLOCK,           | Pointer to event block
: 581      0575 2 FILPTR : REF BITVECTOR,       | Pointer to event filter mask
: 582      0576 2 LOGPTR : REF BITVECTOR,       | Pointer to event log mask
: 583      0577 2 SRCPTR,                   | Pointer to source block
: 584      0578 2 ZERCNT;                  | Trailing zero byte count
: 585      0579 2
: 586      0580 2
: 587      0581 2 | Get the source block.
: 588      0582 2
```

```

589      0583 2    IF NOT NML$MATCHSRC (.DATDSC, .SNK, .SRC, .ENTDSC, SRCPTR)
590      0584 2    THEN
591      0585 2    RETURN FALSE;
592      0586 2
593      0587 2    | Get the event block.
594      0588 2
595      0589 2    IF NOT NML$MATCHEVT (.SRCPTR, .CLASS, EVT PTR)
596      0590 2    THEN
597      0591 2    RETURN FALSE;
598      0592 2
599      0593 2    | Get combined specific and global filters.
600      0594 2
601      0595 2    IF NOT NML$GETCOMFILTERS (.DATDSC, .SNK, .CLASS, .MSKPTR, .RESLEN)
602      0596 2    THEN
603      0597 2    RETURN FALSE;
604      0598 2
605      0599 2    RETURN TRUE
606      0600 2
607      0601 1    END;                                ! End of NML$GETSPCFILTERS

```

			0000 0000	.ENTRY	NML\$GETSPCFILTERS, Save nothing	:	0527
	5E		08 C2 00002	SUBL2	#8, SP		0583
	7E	0C	5E DD 00005	PUSHL	SP		
00000000V	EF	04	AC 7D 00007	MOVQ	SRC, -(SP)		
	2C		05 FB 0000F	MOVQ	DATDSC, -(SP)		
	EF	04	50 E9 00016	CALLS	#5, NML\$MATCHSRC		
	19	14	AE 9F 00019	BLBC	R0, 1\$		
00000000V	EF	08	AC DD 0001C	PUSHAB	EVT PTR		0589
	19	08	AE DD 0001F	PUSHL	CLASS		
	7E	03	03 FB 00022	PUSHL	SRCPTR		
	19	50	E9 00029	CALLS	#3, NML\$MATCHEVT		
	7E	18	AC 7D 0002C	BLBC	R0, 1\$		
	14	14	DD 00030	MOVQ	MSKPTR, -(SP)		0595
00000000V	EF	04	AC 7D 00033	PUSHL	CLASS		
	04	05	FB 00037	MOVQ	DATDSC, -(SP)		
	50	50	E9 0003E	CALLS	#5, NML\$GETCOMFILTERS		
	50	01	D0 00041	BLBC	R0, 1\$		0599
		04	00044	MOVL	#1, R0		
		50	D4 00045 1\$:	RET	RET		
		04	00047	CLRL	R0		0601

; Routine Size: 72 bytes. Routine Base: \$CODE\$ + 0261

```
: 609      0602 1 %SBTTL 'NML$GETCOMFILTERS Get event filters'  
: 610      0603 1 GLOBAL ROUTINE NMLSGETCOMFILTERS (DATDSC, EVTPT, SNK, MSKPTR, RESLEN) =  
: 611      0604 1  
: 612      0605 1 ++  
: 613      0606 1 FUNCTIONAL DESCRIPTION:  
: 614      0607 1  
: 615      0608 1 This routine gets event filters from the specified event block  
: 616      0609 1 and combines them with the global filters for the class. The  
: 617      0610 1 resulting mask is the complete event mask for the class and source.  
: 618      0611 1  
: 619      0612 1 FORMAL PARAMETERS:  
: 620      0613 1  
: 621      0614 1 DATDSC Descriptor of current data base entry.  
: 622      0615 1 EVTPT Pointer to event block.  
: 623      0616 1 SNK Event sink type code.  
: 624      0617 1 MSKPTR Address of filter mask quadword.  
: 625      0618 1 RESLEN Address of longword to contain byte count of  
: 626      0619 1 resulting mask.  
: 627      0620 1  
: 628      0621 1 IMPLICIT INPUTS:  
: 629      0622 1     NONE  
: 630      0623 1  
: 631      0624 1  
: 632      0625 1 IMPLICIT OUTPUTS:  
: 633      0626 1     NONE  
: 634      0627 1  
: 635      0628 1  
: 636      0629 1 ROUTINE VALUE:  
: 637      0630 1 COMPLETION CODES:  
: 638      0631 1  
: 639      0632 1     TRUE is returned if operation is successful. Otherwise, FALSE  
: 640      0633 1 is returned.  
: 641      0634 1  
: 642      0635 1 SIDE EFFECTS:  
: 643      0636 1  
: 644      0637 1     NONE  
: 645      0638 1  
: 646      0639 1 --  
: 647      0640 1  
: 648      0641 2 BEGIN  
: 649      0642 2  
: 650      0643 2 MAP  
: 651      0644 2     DATDSC : REF DESCRIPTOR,  
: 652      0645 2     EVTPT : REF BBLOCK,           ! Pointer to event block  
: 653      0646 2     MSKPTR : REF BITVECTOR;  
: 654      0647 2  
: 655      0648 2 LOCAL  
: 656      0649 2     CLASS,          ! Event class  
: 657      0650 2     FILPTR : REF BITVECTOR,    ! Pointer to event filter mask  
: 658      0651 2     LOGPTR : REF BITVECTOR,    ! Pointer to event log mask  
: 659      0652 2     ZERCNT;          ! Trailing zero byte count  
: 660      0653 2  
: 661      0654 2  
: 662      0655 2     Get global filter mask for this class.  
: 663      0656 2  
: 664      0657 2     CLASS = .EVTPT [EVTSW CLASS];  
: 665      0658 2     NMLSGETGBLFILTERS (.DATDSC, .SNK, .CLASS, .MSKPTR);
```

```

666 0659 2 | Combine specific masks with global mask.
667 0660 2
668 0661 2
669 0662 2 LOGPTR = EVTPTR [EVT$Q_LOGMSK];
670 0663 2 FILPTR = EVTPTR [EVT$Q_FILTERMSK];
671 0664 2
672 0665 2 INCR I FROM 0 TO (EVT$S_LOGMSK * 8) - 1 DO
673 0666 3 BEGIN
674 0667 3
675 0668 3 MSKPTR [.I] = .MSKPTR [.I] OR .LOGPTR [.I];
676 0669 3 MSKPTR [.I] = .MSKPTR [.I] AND NOT .FILPTR [.I];
677 0670 3
678 0671 2 END;
679 0672 2
680 0673 2 | Adjust count to exclude zero bytes on the end of the quadword mask.
681 0674 2
682 0675 2 ZEROCNT = 0;
683 0676 2
684 0677 2 DECR I FROM EVT$S_LOGMSK - 1 DO
685 0678 3 BEGIN
686 0679 3
687 0680 3 IF (.MSKPTR + .I)<0,8> EQLU 0
688 0681 3 THEN
689 0682 3 ZEROCNT = .ZEROCNT + 1
690 0683 3 ELSE
691 0684 3 EXITLOOP;
692 0685 3
693 0686 2 END;
694 0687 2
695 0688 2 | Set up mask length for return.
696 0689 2
697 0690 2 .RESLEN = EVT$S_LOGMSK - .ZEROCNT;
698 0691 2
699 0692 2 RETURN TRUE
700 0693 2
701 0694 1 END;

```

				003C	00000	.ENTRY	NML\$GETCOMFILTERS, Save R2,R3,R4,R5	: 0603
		50	08	BC	3C 00002	MOVZWL	AEVTPTR, CLASS	: 0657
		53	10	AC	DO 00006	MOVL	MSKPTR, R3	: 0658
				09	BB 0000A	PUSHR	#^M<R0,R3>	
			0C	AC	DD 0000C	PUSHL	SNK	
			04	AC	DD 0000F	PUSHL	DATDSC	
		00000000V	EF	04	FB 00012	CALLS	#4, NML\$GETGBLFILTERS	
	54	08	AC	04	C1 00019	ADDL3	#4, EVTPTR, LOGPTR	: 0662
	55	08	AC	0C	C1 0001E	ADDL3	#12, EVTPTR, FILPTR	: 0663
				51	D4 00023	CLRL	I	: 0665
52	63		01	51	EF 00025	EXTZV	I, #1, (R3), R2	: 0668
50	64		01	51	EF 0002A	EXTZV	I, #1, (LOGPTR), R0	
			50	52	C8 0002F	BISL2	R2, R0	
63	01		51	50	F0 00032	INSV	R0, I, #1, (R3)	
52	63		01	51	EF 00037	EXTZV	I, #1, (R3), R2	: 0669
50	65		01	51	EF 0003C	EXTZV	I, #1, (FILPTR), R0	

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$GETCOMFILTERS Get event filters

B 10
16-Sep-1984 00:19:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:50:11 [NML.SRC]NMLLOGOPS.B32;1

Page 23
(8)

63	01	52	50	CA	00041	BICL2	R0, R2	
	D8	51	52	F0	00044	INSV	R2, I, #1\$, (R3)	0665
		51	3F	F3	00049	AOBLEQ	#63, I, 1\$	0677
		50	07	7D	0004D	MOVQ	#7, I	0680
			6043	95	00050	2\$: TSTB	(I\$[R3])	0682
			05	12	00053	BNEQ	3\$	0677
14	BC	F6	51	D6	00055	INCL	ZERCNT	0690
		08	50	F4	00057	S0BGEQ	I, 2\$	0692
		50	51	C3	0005A	3\$: SUBL3	ZERCNT, #8, @RESLEN	0694
			01	D0	0005F	MOVL	#1, R0	
			04	00062		RET		

; Routine Size: 99 bytes, Routine Base: \$CODE\$ + 02A9

```
703      0695 1 %SBTTL 'NML$GETGBLFILTERS Get global filters for sink and class'
704      0696 1 GLOBAL ROUTINE NML$GETGBLFILTERS (DATDSC, SNK, CLASS, MSKPTR) =
705      0697 1
706      0698 1 ++
707      0699 1 | FUNCTIONAL DESCRIPTION:
708      0700 1 |
709      0701 1 | This routine returns the global filters for the specified
710      0702 1 | sink type and class.
711      0703 1
712      0704 1 | FORMAL PARAMETERS:
713      0705 1
714      0706 1 | DATDSC      Descriptor of source block buffer.
715      0707 1 | SNK         Logging sink type code.
716      0708 1 | CLASS        Event class code.
717      0709 1 | MSKPTR      Pointer to quadwrd to contain global filter mask.
718      0710 1
719      0711 1 | IMPLICIT INPUTS:
720      0712 1 |     NONE
721      0713 1
722      0714 1
723      0715 1 | IMPLICIT OUTPUTS:
724      0716 1 |     NONE
725      0717 1
726      0718 1
727      0719 1 | ROUTINE VALUE:
728      0720 1 | COMPLETION CODES:
729      0721 1
730      0722 1 | TRUE is returned if global filters are found, FALSE is returned
731      0723 1 | if no global filters are found. If no global filters are found
732      0724 1 | the resulting filter mask will be zeroed.
733      0725 1
734      0726 1 | SIDE EFFECTS:
735      0727 1
736      0728 1 |     NONE
737      0729 1
738      0730 1 | --
739      0731 1 | BEGIN
740      0732 2
741      0733 2
742      0734 2 | LOCAL
743      0735 2 |     EVT PTR : REF BBLOCK,          | Event block pointer
744      0736 2 |     SRC PTR : REF BBLOCK,          | Source block pointer
745      0737 2 |     STATUS:                      | Routine status
746      0738 2
747      0739 2 | Zero the filter mask.
748      0740 2
749      0741 2 | CH$FILL (0, EVT$$_LOGMSK, .MSKPTR);
750      0742 2
751      0743 2 | If global filters are found then just return.
752      0744 2
753      0745 2 | IF NOT NML$MATCHSRC (.DATDSC,
754      0746 2 |     .SNK,
755      0747 2 |     NMASC$_ENT_KNO,
756      0748 2 |     UPLIT(0,-0),
757      0749 2 |     SRCPTR)
758      0750 2 | THEN
759      0751 2 |     RETURN FALSE;
```

```

760 0752 2
761 0753 2
762 0754 2 | If global filters are found for the specified class then move them
763 0755 2 into the result mask.
764 0756 2
765 0757 2 IF NML$MATCHEVT (.SRCPTR,
766 0758 2 .CLASS
767 0759 2 EVTPTRS)
768 0760 2 THEN
769 0761 2 BEGIN
770 0762 2
771 0763 3 CH$MOVE (EVT$S LOGMSK,
772 0764 3 EVTPTR [EVT$Q_LOGMSK],
773 0765 3 MSKPTR);
774 0766 3 STATUS = TRUE;
775 0767 3
776 0768 3
777 0769 2 ELSE
778 0770 2 STATUS = FALSE;
779 0771 2
780 0772 2 RETURN .STATUS
781 0773 2
782 0774 1 END;           ! End of NML$GETGBLFILTERS

```

.PSECT \$SPLIT\$,NOWRT,NOEXE,2

00000000 00000000 00020 P.AAE: .LONG 0, 0

.PSECT \$CODE\$,NOWRT,2

			003C 00000	.ENTRY NML\$GETGBLFILTERS, Save R2,R3,R4,R5	0696
08	00	5E	08 C2 00002	SUBL2 #8, SP	0741
		6E	00 2C 00005	MOV C5 #0, (SP), #0, #8, @MSKPTR	
			0000A	PUSHL SP	0745
			BC	PUSHAB P.AAE	0748
			10	MNEG L #1, -(SP)	0745
			5E	MOV Q DATA DSC, -(SP)	
			DD	CALLS #5, NML\$MATCHSRC	
			00000000	BLBC R0, 1\$	
			'	PUSHAB EVT PTR	0757
			EF	AC 7D 00014	0758
			01	MNEG L #1, -(SP)	0757
			CE	MOV Q DATA DSC, -(SP)	
			04	CALLS #5, NML\$MATCHSRC	
			AC	BLBC R0, 1\$	
			05	PUSHL CLASS	0757
			FB	SRC PTR	0757
			0001B	CALLS #3, NML\$MATCHEVT	
			50	BLBC R0, 1\$	
			E9	MOVL EVT PTR, R0	0764
			00022	MOV C3 #8, 4(R0), @MSKPTR	0765
			AE	MOVL #1, STATUS	0766
			9F	RET	0757
			00025	CLRL R0	0774
			0C		
			AC		
			00028		
			08		
			AE		
			DD		
			0002B		
			03		
			FB		
			0002E		
			50		
			E9		
			00035		
			AE		
			DO		
			00038		
			08		
			28		
			0003C		
			01		
			DO		
			00042		
			04		
			00045		
			50		
			D4		
			00046	1\$:	
			04		
			00048		

; Routine Size: 73 bytes, Routine Base: \$CODE\$ + 030C

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$GETGBLFILTERS Get global filters for sink

E 10

16-Sep-1984 00:19:25

14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 26
(9)

NM
VO

```
: 784      0775 1 %SBTTL 'NML$CLEANEVT Clean event masks'  
: 785      0776 1 GLOBAL ROUTINE NML$CLEANEVT (SNK, BLKDSC) : NOVALUE =  
: 786      0777 1  
: 787      0778 1 ++  
: 788      0779 1 | FUNCTIONAL DESCRIPTION:  
: 789      0780 1 | This routine runs through all sources for the specified sink type  
: 790      0781 1 | and deletes all event filters that match the global filters.  
: 791      0782 1  
: 792      0783 1  
: 793      0784 1 | FORMAL PARAMETERS:  
: 794      0785 1  
: 795      0786 1 | SNK          Logging sink type code.  
: 796      0787 1 | BLKDSC       Descriptor of all source block data.  
: 797      0788 1  
: 798      0789 1 | IMPLICIT INPUTS:  
: 799      0790 1  
: 800      0791 1 |     NONE  
: 801      0792 1  
: 802      0793 1 | IMPLICIT OUTPUTS:  
: 803      0794 1  
: 804      0795 1 |     NONE  
: 805      0796 1  
: 806      0797 1 | ROUTINE VALUE:  
: 807      0798 1 | COMPLETION CODES:  
: 808      0799 1  
: 809      0800 1 |     NONE  
: 810      0801 1  
: 811      0802 1 | SIDE EFFECTS:  
: 812      0803 1  
: 813      0804 1 |     NONE  
: 814      0805 1  
: 815      0806 1 |--  
: 816      0807 1  
: 817      0808 2 | BEGIN  
: 818      0809 2  
: 819      0810 2 | LOCAL  
: 820      0811 2 |     EVTPTR : REF BBLOCK,  
: 821      0812 2 |     FILMSK : REF BITVECTOR,  
: 822      0813 2 |     GBLEVT : REF BBLOCK,  
: 823      0814 2 |     GBLMSK : REF BITVECTOR,  
: 824      0815 2 |     LOGMSK : REF BITVECTOR,  
: 825      0816 2 |     GBLSRC : REF BBLOCK,  
: 826      0817 2 |     SRCPTR : REF BBLOCK,  
: 827      0818 2 |     STATUS;  
: 828      0819 2  
: 829      0820 2 | If there are no global filters then just clean up the filter masks.  
: 830      0821 2  
: 831      0822 2 | IF NML$MATCHSRC (.BLKDSC, .SNK, NMASC_ENT_KNO, 0, GBLSRC)  
: 832      0823 2 | THEN  
: 833      0824 3 | BEGIN  
: 834      0825 3  
: 835      0826 3 | Make sure the filter mask is zeroed for the global filters.  
: 836      0827 3  
: 837      0828 3 |     GBLEVT = 0;  
: 838      0829 3 | WHILE NML$GETNXTEVT (.GBLSRC, GBLEVT) DO  
: 839      0830 4 |     BEGIN  
: 840      0831 4
```

```
: 841      0832 4      GBLMSK = GBLEVT [EVT$Q_FILTERMSK];
: 842      0833 4
: 843      0834 4      INCR I FROM 0 TO (EVT$S_FILTERMSK * 8) - 1 DO
: 844      0835 5      BEGIN
: 845      0836 5
: 846      0837 5      GBLMSK [.I] = 0;
: 847      0838 5
: 848      0839 4      END;
: 849      0840 3      END;
: 850      0841 3      END
: 851      0842 2      ELSE
: 852      0843 2      GBLSRC = 0;
: 853      0844 2
: 854      0845 2      | For every source clean up all event masks.
: 855      0846 2
: 856      0847 2      SRCPTR = 0;
: 857      0848 2      WHILE NML$GETNXTSNK (.BLKDSC, .SNK, SRCPTR) DO
: 858      0849 3      BEGIN
: 859      0850 3      IF .(SRCPTR [SRC$B_SRCTYPE])<0,8,1> NEQ NMASC_ENT_KNO
: 860      0851 3      THEN
: 861      0852 4      BEGIN
: 862      0853 4
: 863      0854 4      | For every event mask get rid of everything that matches the global
: 864      0855 4      filters.
: 865      0856 4
: 866      0857 4      EVTPTR = 0;
: 867      0858 4      WHILE NML$GETNXT_EVT (.SRCPTR, EVTPTR) DO
: 868      0859 5      BEGIN
: 869      0860 5
: 870      0861 5      LOGMSK = EVTPTR [EVT$Q_LOGMSK];
: 871      0862 5      FILMSK = EVTPTR [EVT$Q_FILTERMSK];
: 872      0863 5
: 873      0864 5      IF .GBLSRC NEQA 0
: 874      0865 5      THEN
: 875      0866 5      STATUS = NML$MATCHEVT (.GBLSRC,
: 876      0867 5          .EVTPTR [EVT$W_CLASS],
: 877      0868 5          GBLEVT)
: 878      0869 5      ELSE
: 879      0870 5      STATUS = FALSE;
: 880      0871 5
: 881      0872 5      IF .STATUS
: 882      0873 6      AND (.GBLSRC NEQA 0)
: 883      0874 5      THEN
: 884      0875 6      BEGIN
: 885      0876 6
: 886      0877 6      GBLMSK = GBLEVT [EVT$Q_LOGMSK];
: 887      0878 6
: 888      0879 6      INCR I FROM 0 TO (EVT$S_LOGMSK * 8) - 1 DO
: 889      0880 7      BEGIN
: 890      0881 7
: 891      0882 7      LOGMSK [.I] = .LOGMSK [.I] AND NOT .GBLMSK [.I];
: 892      0883 7      FILMSK [.I] = .FILMSK [.I] AND .GBLMSK [.I];
: 893      0884 7
: 894      0885 6      END;
: 895      0886 6
: 896      0887 5      ELSE
: 897      0888 6      BEGIN
```

```
898      0889 6           INCR I FROM 0 TO (EVT$$_LOGMSK * 8) - 1 DC
899      0890 7           BEGIN
900      0891 7
901      0892 7           FILMSK [.I] = 0;
902      0893 7
903      0894 6           END;
904      0895 5           END;
905      0896 4           END;
906      0897 3           END;
907      0898 2           END;
908      0899 2           END;
909      0900 1           END;               ! End of NML$CLEANEVT
```

			03FC	00000	.ENTRY	NML\$CLEANEVT, Save R2,R3,R4,R5,R6,R7,R8,R9	0776	
59	00000000V	EF	9E	00002	MOVAB	NML\$GETNXTEVT, R9		
5E		10	C2	00009	SUBL2	#16, SP	0822	
		5E	DD	0000C	PUSHL	SP		
		7E	D4	0000E	CLRL	-(SP)		
7E		01	CE	00010	MNEGL	#1, -(SP)		
		04	AC	DD 00013	PUSHL	SNK		
00000000V	EF	08	AC	DD 00016	PUSHL	BLKDSC		
20		05	FB	00019	CALLS	#5, NML\$MATCHSRC		
		50	E9	00020	BLBC	R0, 4\$		
		0C	AE	D4 00023	CLRL	GBLEVT	0828	
		0C	AE	9F 00026	1\$: PUSHAB	GBLEVT	0829	
		04	AE	DD 00029	PUSHL	GBLSRC		
		69	02	FB 0002C	CALLS	#2, NML\$GETNXTEVT		
		13	50	E9 0002F	BLBC	R0, 5\$		
54	OC	AE	OC	C1 00032	ADDL3	#12, GBLEVT, GBLMSK	0832	
			50	D4 00037	CLRL	I	0837	
00		64	50	E5 00039	2\$: BBCC	I, (GBLMSK), 3\$		
F8		50	3F	F3 0003D	3\$: AOBLEQ	#63, I, 2\$	0834	
			E3	11 00041	BRB	1\$	0829	
			6E	D4 00043	4\$: CLRL	GBLSRC	0843	
		04	AE	D4 00045	5\$: CLRL	SRCPTR	0847	
		04	AE	9F 00048	6\$: PUSHAB	SRCPTR	0848	
			04	AC	DD 0004B	PUSHL	SNK	
00000000V	EF	08	AC	DD 0004E	PUSHL	BLKDSC		
01		03	FB	00051	CALLS	#3, NML\$GETNXTSNK		
		50	E8	00058	BLBS	R0, 7\$		
			04	0005B	RET			
FF	53	04	AE	D0 0005C	7\$: MOVL	SRCPTR, R3	0850	
8F		03	A3	91 00060	CMPB	3(R3), #-1		
			E1	13 00065	BEQL	6\$		
		08	AE	D4 00067	CLRL	EVTPTR	0857	
		08	AE	9F 0006A	8\$: PUSHAB	EVTPTR	0858	
			53	DD 0006D	PUSHL	R3		
			02	FB 0006F	CALLS	#2, NML\$GETNXTEVT		
56	69		50	E9 00072	BLBC	R0, 6\$		
55	D3		04	C1 00075	ADDL3	#4, EVTPTR, LOGMSK	0861	
	08	AE	0C	C1 0007A	ADDL3	#12, EVTPTR, FILMSK	0862	
	08	AE	52	D4 0007F	CLRL	R2	0864	
			6E	D5 00081	TSTL	GBLSRC		

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$CLEANEVT [Clean event masks]

I 10

16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 30
(10)

				18	13	00083		BEQL	9\$		
				52	D6	00085		INCL	R2		
				OC	AE	9F	00087	PUSHAB	GBLEVT		
				OC	BE	3C	0008A	MOVZWL	AEVT PTR, -(SP)		
				08	AE	DD	0008E	PUSHL	GBLSRC		
			00000000V	7E	03	FB	00091	CALLS	#3, NML\$MATCHEV		
					50	DO	00098	MOVL	R0, STATUS		
					02	11	0009B	BRB	10\$		
					57	D4	0009D	9\$: CLRL	STATUS		0870
					57	E9	0009F	10\$: BLBC	STATUS, 12\$		0872
				54	OC	37	000A2	BLBC	R2, 12\$		0873
					34	AE	000A5	ADDL3	#4, GBLEVT, GBLMSK		0877
							50	CLRL	I		0882
52	66				01	50	EF 000AC	11\$: EXTZV	I, #1, (LOGMSK), R2		
51	64				01	50	EF 000B1	EXTZV	I, #1, (GBLMSK), R1		
					52	CA	000B6	BICL2	R1, R2		
66	01				50	F0	000B9	INSV	R2, I, #1, (LOGMSK)		
52	65				01	50	EF 000BE	EXTZV	I, #1, (FILMSK), R2		0883
51	64				58	EF	000C3	EXTZV	I, #1, (GBLMSK), R1		
					52	D2	000C8	MCOML	R2, R8		
					51	CA	000CB	BICL2	R8, R1		
65	01				50	F0	000CE	INSV	R1, I, #1, (FILMSK)		
	D5				50	F3	000D3	AOBLEQ	#63, I, 11\$		0879
					91	11	000D7	BRB	8\$		0872
					50	D4	000D9	12\$: CLRL	I		0889
	00				65	50	E5 000DB	13\$: BBCC	I, (FILMSK), 14\$		0892
	F8				50	F3	000DF	14\$: AOBLEQ	#63, I, 13\$		0889
					85	11	000E3	BRB	8\$		0858
							04	RET			0900

: Routine Size: 230 bytes, Routine Base: \$CODE\$ + 0355

```

911      0901 1 %SBTTL 'NML$CLEANSRC Clean sources'
912      0902 1 GLOBAL ROUTINE NML$CLEANSRC (BUFDSC, SNK, BLKDSC) : NOVALUE =
913      0903 1
914      0904 1 ++
915      0905 1 | FUNCTIONAL DESCRIPTION:
916      0906 1
917      0907 1 | This routine goes through all source blocks for the specified
918      0908 1 | sink type and removes all event blocks that have no filters set.
919      0909 1 | Source blocks with event blocks are also removed.
920      0910 1
921      0911 1 | FORMAL PARAMETERS:
922      0912 1
923      0913 1 | BUFDSC      Descriptor of buffer containing source blocks.
924      0914 1 | SNK         Logging sink type code.
925      0915 1 | BLKDSC     Descriptor of all source block data in buffer.
926      0916 1
927      0917 1 | IMPLICIT INPUTS:
928      0918 1 |     NONE
929      0919 1
930      0920 1 | IMPLICIT OUTPUTS:
931      0921 1 |     NONE
932      0922 1
933      0923 1
934      0924 1
935      0925 1 | ROUTINE VALUE:
936      0926 1 | COMPLETION CODES:
937      0927 1 |     NONE
938      0928 1
939      0929 1
940      0930 1 | SIDE EFFECTS:
941      0931 1 |     NONE
942      0932 1
943      0933 1
944      0934 1 |
945      0935 1
946      0936 2 | BEGIN
947      0937 2
948      0938 2 | LOCAL
949      0939 2 |     EVT PTR : REF BBLOCK,          ! Pointer to event block
950      0940 2 |     FILMSK : REF BITVECTOR,
951      0941 2 |     LOGMSK : REF BITVECTOR,
952      0942 2 |     OLDEVT : REF BBLOCK,          ! Pointer to previous event block
953      0943 2 |     OLDSRC : REF BBLOCK,          ! Pointer to previous source block
954      0944 2 |     SRCPTR : REF BBLOCK,          ! Pointer to current source block
955      0945 2 |     STATUS;
956      0946 2
957      0947 2 |     OLDSRC = 0;
958      0948 2 |     SRCPTR = 0;
959      0949 2 |     WHILE NML$GETNXTSNK (.BLKDSC, .SNK, SRCPTR) DO
960      0950 3 |     BEGIN
961      0951 3 |     CHSMOVE (.SRCPTR [SRC$W_LENGTH], .SRCPTR, NML$T_SRCBUFFER);
962      0952 3
963      0953 3
964      0954 3 |     OLDEVT = 0;
965      0955 3 |     EVT PTR = 0;
966      0956 3 |     WHILE NML$GETNXT_EVT (NML$T_SRCBUFFER, EVT PTR) DO
967      0957 4 |     BEGIN

```

```

968      0958  4
969      0959  4
970      0960  4
971      0961  4
972      0962  4
973      0963  4
974      0964  5
975      0965  5
976      0966  5
977      0967  5
978      0968  6
979      0969  6
980      0970  6
981      0971  5
982      0972  4
983      0973  4
984      0974  4
985      0975  4
986      0976  5
987      0977  5
988      0978  5
989      0979  5
990      0980  4
991      0981  4
992      0982  4
993      0983  3
994      0984  3
995      0985  3
996      0986  3
997      0987  4
998      0988  4
999      0989  4
1000     0990  4
1001     0991  3
1002     0992  4
1003     0993  4
1004     0994  4
1005     0995  3
1006     0996  3
1007     0997  2
1008     0998  2
1009     0999  1

      LOGMSK = EVT PTR [EVT$Q_LOGMSK];
      FILMSK = EVT PTR [EVT$Q_FILTERMSK];
      STATUS = FALSE;
      INCR I FROM 0 TO (EVT$S_LOGMSK * 8) - 1 DO
        BEGIN
          IF .LOGMSK [.I] OR .FILMSK [.I]
            THEN
              BEGIN
                STATUS = TRUE;
                EXITLOOP;
              END;
        END;

        IF NOT .STATUS
        THEN
          BEGIN
            NML$REMEVT (NML$T_SRCBUFFER, .EVT PTR);
            EVT PTR = .OLDEVT; ! Back up event pointer
          END
        ELSE
          OLDEVT = .EVT PTR;

        END;

        IF .NML$T_SRCBUFFER [SRC$W_MSKCOUNT] NEQU 0
        THEN
          BEGIN
            NML$REPSRC (.BUFDSC, .BLKDSC, .SRCPTR, NML$T_SRCBUFFER);
            OLDSRC = .SRCPTR;
          END
        ELSE
          BEGIN
            NML$REMSRC (.BLKDSC, .SRCPTR);
            SRCPTR = .OLDSRC; ! Back up the source pointer
          END;
        END;
      END;
    END;
  ! End of NML$CLEANSRC

```

		OFFC 00000	.ENTRY	NML\$CLEANSRC, Save R2,R3,R4,R5,R6,R7,R8,R9,-; 0902
	5E	04 C2 00002	R10,R11	
		58 D4 00005	#4, SP	
		7E D4 00007	CLRL OLDSRC	0947
		5E DD 00009 1\$:	CLRL SRCPTR	0948
	08	AC DD 0000B	PUSHL SP	0949
	0C	AC DD 0000E	PUSHL SNK	
00000000V	EF	03 FB 00011	PUSHL BLKDSC	
	01	50 E8 00018	CALLS #3, NML\$GETNXTSNK	
			BLBS R0, 2\$	

NML SLOGOPS
V04-000

NML Logging data base operations module
NML\$CLEANSRC Clean sources

L 10
16-Sep-1984 00:19:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:50:11 [NML.SRC]NMLLOGOPS.B32;1

Page 33
(11)

NM
VO

00000000'	EF	56	6E	04	0001B	RET		0952	
		66	66	00	0001C	2\$: MOVL	SRCPTR, R6	0954	
			66	28	0001F	MOV C3	(R6), (R6), NMLST_SRCBUFFER	0955	
			59	D4	00027	CLRL	OLDEVT	0956	
			AE	D4	00029	CLRL	EVT PTR		
			AE	9F	0002C	3\$: PUSHAB	EVT PTR		
		04	FF	9F	0002F	PUSHAB	NMLST_SRCBUFFER		
		04	02	FB	00035	CALLS	#2, NMLSGETNXTEVT		
		00000000V	EF	50	E9	0003C	BLBC	R0, 9\$	
5A	04	AE	04	C1	0003F	ADDL3	#4, EVT PTR, LOGMSK	0959	
5B	04	AE	OC	C1	00044	ADDL3	#12, EVT PTR, FILMSK	0960	
			57	D4	00049	CLRL	STATUS	0962	
			50	D4	0004B	CLRL	I	0966	
04		6A	50	E0	0004D	4\$: BBS	I, (LOGMSK), 5\$		
05		6B	50	E1	00051	BBC	I, (FILMSK), 6\$		
		57	01	D0	00055	5\$: MOVL	#1, STATUS	0969	
			04	11	00058	BRB	7\$	0968	
EF		50	3F	F3	0005A	6\$: AOBLEQ	#63, I, 4\$	0963	
		16	57	E8	0005E	7\$: BLBS	STATUS, 8\$	0974	
			04	DD	00061	PUSHL	EVT PTR	0977	
		00000000V	EF	9F	00064	PUSHAB	NMLST_SRCBUFFER		
		04	02	FB	0006A	CALLS	#2, NMLSREMEVT		
		AE	59	D0	00071	MOVL	OLDEVT, EVT PTR	0978	
			B5	11	00075	BRB	3\$	0974	
		59	04	AE	D0	00077	8\$: MOVL	EVT PTR, OLDEVT	0981
				AF	11	0007B	BRB	3\$	0956
			00000000'	EF	B5	0007D	9\$: TSTW	NMLST_SRCBUFFER+22	0985
				1A	13	00083	BEQL	10\$	
			00000000'	EF	9F	00085	PUSHAB	NMLST_SRCBUFFER	0988
				56	DD	0008B	PUSHL	R6	
				OC	AC	0008D	PUSHL	BLKDSC	
				04	AC	00090	PUSHL	BUFDSC	
		00000000V	EF	04	FB	00093	CALLS	#4, NMLSREPSRC	
		58	56	D0	0009A	MOVL	R6, OLDSRC	0989	
			0F	11	0009D	BRB	11\$	0985	
			56	DD	0009F	10\$: PUSHL	R6	0993	
			OC	AC	DD	000A1	PUSHL	BLKDSC	
		00000000V	EF	02	FB	000A4	CALLS	#2, NMLSREMSRC	
		6E	58	D0	000AB	MOVL	OLDSRC, SRCPTR	0994	
			FF	31	000AE	11\$: BRW	1\$	0949	
				04	000B1	RET		0999	

Routine Size: 178 bytes. Routine Base: \$CODES + 043B

```
: 1011    1000 1 %SBTTL 'NML$MATCHSRC Match specific source'  
: 1012    1001 1 GLOBAL ROUTINE NML$MATCHSRC (BLKDSC, SNK, SRC, ENTDSC, SRCPTR) =  
: 1013    1002 1  
: 1014    1003 1 ++  
: 1015    1004 1 | FUNCTIONAL DESCRIPTION:  
: 1016    1005 1 |  
: 1017    1006 1 | This routine searches the sink node buffer for a source block  
: 1018    1007 1 | that matches the specified event source.  
: 1019    1008 1  
: 1020    1009 1 | FORMAL PARAMETERS:  
: 1021    1010 1  
: 1022    1011 1 | BLKDSC      Descriptor of source block buffer.  
: 1023    1012 1 | SNK          Logging sink type code.  
: 1024    1013 1 | SRC          Event source type code.  
: 1025    1014 1 | ENTDSC      Event source id string descriptor.  
: 1026    1015 1 | SRCPTR      Pointer to longword in which to return address  
: 1027    1016 1 | of source block.  
: 1028    1017 1  
: 1029    1018 1 | IMPLICIT INPUTS:  
: 1030    1019 1 |     NONE  
: 1031    1020 1 | IMPLICIT OUTPUTS:  
: 1032    1021 1 |     NONE  
: 1033    1022 1 | ROUTINE VALUE:  
: 1034    1023 1 | COMPLETION CODES:  
: 1035    1024 1 |  
: 1036    1025 1 |  
: 1037    1026 1 |  
: 1038    1027 1 |  
: 1039    1028 1 |  
: 1040    1029 1 |  
: 1041    1030 1 |  
: 1042    1031 1 |  
: 1043    1032 1 |  
: 1044    1033 1 |  
: 1045    1034 1 |  
: 1046    1035 1 |--  
: 1047    1036 1  
: 1048    1037 2 | BEGIN  
: 1049    1038 2  
: 1050    1039 2 | MAP  
: 1051    1040 2 |     SRC : BYTE,  
: 1052    1041 2 |     ENTDSC : REF DESCRIPTOR;  
: 1053    1042 2 | LOCAL  
: 1054    1043 2 |     PTR : REF BBLOCK,           | Temporary source block pointer  
: 1055    1044 2 |     STATUS,                  | Routine status  
: 1056    1045 2 |     TSTLEN,                  | Length of source to compare  
: 1057    1046 2 |     TSTPTR;                 | Address of source to compare  
: 1058    1047 2  
: 1059    1048 2  
: 1060    1049 2 |     PTR = 0;                | Initialize source pointer  
: 1061    1050 2 |     STATUS = FALSE;          | Initialize routine status  
: 1062    1051 2  
: 1063    1052 2 | WHILE NML$GETNXTSNK (.BLKDSC, .SNK, PTR) DO  
: 1064    1053 2 | BEGIN  
: 1065    1054 3 |     IF .PTR [SRC$B_SRCTYPE] EQLU .SRC  
: 1066    1055 3 |     THEN  
: 1067    1056 4 |         BEGIN
```

```
: 1068      1057 4 |  
: 1069      1058 4 | Select the length and address of the source to compare.  
: 1070      1059 4 |  
: 1071      1060 4 |     SELECTONEU .SRC OF  
: 1072      1061 4 |     SET  
: 1073      1062 4 |  
: 1074      1063 4 |     [NMASC_ENT_NOD]:      ! Node  
: 1075      1064 5 |     BEGIN  
: 1076      1065 5 |  
: 1077      1066 5 |     IF .(ENTDSC [DSC$A_POINTER])<0,16> EQLU  
: 1078      1067 5 |     .PTR [SRC$W_NODADR]  
: 1079      1068 5 |     THEN STATUS = TRUE;  
: 1080      1069 5 |  
: 1081      1070 5 |  
: 1082      1071 4 |     END;  
: 1083      1072 4 |  
: 1084      1073 4 |     [NMASC_ENT_CIR,  
: 1085      1074 4 |     NMASC_ENT_LIN  
: 1086      1075 4 |     NMASC_ENT_MOD]:      ! Circuit or Line or Module  
: 1087      1076 5 |     BEGIN  
: 1088      1077 5 |  
: 1089      1078 5 |     IF CHSEQL (.ENTDSC [DSC$W_LENGTH],  
: 1090      1079 5 |     .ENTDSC [DSC$A_POINTER],  
: 1091      1080 5 |     .PTR [SRC$B_ID[LENGTH],  
: 1092      1081 5 |     PTR [SRC$T_ID])  
: 1093      1082 5 |     THEN STATUS = TRUE;  
: 1094      1083 5 |  
: 1095      1084 5 |  
: 1096      1085 4 |     END;  
: 1097      1086 4 |  
: 1098      1087 4 |     [OTHERWISE]:      ! Null  
: 1099      1088 5 |     BEGIN  
: 1100      1089 5 |  
: 1101      1090 5 |     STATUS = TRUE;  
: 1102      1091 5 |  
: 1103      1092 4 |     END;  
: 1104      1093 4 |     TES;  
: 1105      1094 4 |  
: 1106      1095 4 |     IF .STATUS  
: 1107      1096 4 |     THEN BEGIN  
: 1108      1097 5 |  
: 1109      1098 5 |  
: 1110      1099 5 |     .SRCPTR = .PTR;  
: 1111      1100 5 |     EXITLOOP;  
: 1112      1101 5 |  
: 1113      1102 4 |     END;  
: 1114      1103 3 |     END;  
: 1115      1104 2 |     END;  
: 1116      1105 2 |  
: 1117      1106 2 |     RETURN .STATUS  
: 1118      1107 2 |  
: 1119      1108 1 |     END;          ! End of NML$MATCHSRC
```

NML\$LOGOPS
V04-000NML Logging data base operations module
NML\$MATCHSRC Match specific sourceB 11
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11
VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1Page 36
(12)NML
V04

				007C 00000	ENTRY	NML\$MATCHSRC, Save R2,R3,R4,R5,R6	
				7E D4 00002	CLRL	PTR	: 1001
				56 D4 00004	CLRL	STATUS	: 1049
			55 0C	AC 9A 00006	MOVZBL	SRC, R5	: 1050
				SE DD 0000A	PUSHL	SP	: 1054
			7E 04	AC 7D 0000C	MOVQ	BLKDSC, -(SP)	: 1052
				EF 03	CALLS	#3, NML\$GETNXTSNK	: 1
			45	50 E9 00017	BLBC	R0, 7\$: 1
			54	6E D0 0001A	MOVL	PTR, R4	: 1
			08	00 ED 0001D	CMPZV	#0, #8, 3(R4), R5	: 1054
				E5 12 00023	BNEQ	1\$: 1
				55 D5 00025	TSTL	R5	: 1063
				0B 12 00027	BNEQ	2\$: 1
			04 50	AC D0 00029	MOVL	ENTDSC, R0	: 1066
			A4 04	A0 B1 0002D	CMPW	4(R0), 4(R4)	: 1067
				1F 11 00032	BRB	4\$: 1
			01	55 91 00034	CMPB	R5, #1	: 1073
				0A 13 00037	BEQL	3\$: 1
			03	55 91 00039	CMPB	R5, #3	: 1
				17 1F 0003C	BLSSU	5\$: 1
			04	55 91 0003E	CMPB	R5, #4	: 1
				12 1A 00041	BGTRU	5\$: 1
			50 10	AC D0 00043	MOVL	ENTDSC, R0	: 1078
			51 04	A4 9A 00047	MOVZBL	4(R4), R1	: 1080
			B0 05	60 2D 0004B	CMPCS	(R0), @4(R0), #0, R1, 5(R4)	: 1081
				A4 00051	BNEQ	6\$: 1
				03 12 00053	MOVL	#1, STATUS	: 1090
			56 AF	01 D0 00055	BLBC	STATUS, 1\$: 1095
			14 BC	56 E9 00053	MOVL	R4, @SRCPTR	: 1099
			50	54 D0 0005B	MOVL	STATUS, R0	: 1106
				56 D0 0005F	RET		: 1108
				04 00062			: 1

: Routine Size: 99 bytes, Routine Base: \$CODE\$ + 04ED

```
: 1121    1109 1 %SBTTL 'NML$GETNXTSNK Get next source block for specified sink'  
: 1122    1110 1 GLOBAL ROUTINE NML$GETNXTSNK (BLKDSC, SNK, SRCPTR) =  
: 1123    1111 1 !++  
: 1124    1112 1 | FUNCTIONAL DESCRIPTION:  
: 1125    1113 1 | This routine searches the sink node buffer for the next source block  
: 1126    1114 1 | that matches the specified sink type.  
: 1127    1115 1 |  
: 1128    1116 1 | FORMAL PARAMETERS:  
: 1129    1117 1 |  
: 1130    1118 1 |  
: 1131    1119 1 |  
: 1132    1120 1 | BLKDSC Descriptor of event source block buffer.  
: 1133    1121 1 | SNK Logging sink type code to match.  
: 1134    1122 1 | SRCPTR Address of longword in which to return address  
: 1135    1123 1 | of source block. If within range of buffer  
: 1136    1124 1 | it will be used as the starting point from which  
: 1137    1125 1 | to get the next source block that matches the  
: 1138    1126 1 | specified sink.  
: 1139    1127 1 |  
: 1140    1128 1 | IMPLICIT INPUTS:  
: 1141    1129 1 |  
: 1142    1130 1 |     NONE  
: 1143    1131 1 |  
: 1144    1132 1 | IMPLICIT OUTPUTS:  
: 1145    1133 1 |  
: 1146    1134 1 |     NONE  
: 1147    1135 1 |  
: 1148    1136 1 | ROUTINE VALUE:  
: 1149    1137 1 | COMPLETION CODES:  
: 1150    1138 1 |  
: 1151    1139 1 |     TRUE is returned if a match is found, FALSE is returned if no match.  
: 1152    1140 1 |  
: 1153    1141 1 | SIDE EFFECTS:  
: 1154    1142 1 |  
: 1155    1143 1 |     NONE  
: 1156    1144 1 |  
: 1157    1145 1 | ---  
: 1158    1146 1 |  
: 1159    1147 2 | BEGIN  
: 1160    1148 2 |  
: 1161    1149 2 | LOCAL  
: 1162    1150 2 |     PTR : REF BBLOCK,          ! Temporary source block pointer  
: 1163    1151 2 |     STATUS;           ! Routine status  
: 1164    1152 2 |  
: 1165    1153 2 |     STATUS = FALSE;        ! Initialize routine status  
: 1166    1154 2 |     PTR = ..SRCPTR;       ! Initialize source pointer  
: 1167    1155 2 |  
: 1168    1156 2 | WHILE NML$GETNXTSRC (.BLKDSC, PTR) DO  
: 1169    1157 3 |     BEGIN  
: 1170    1158 3 |     IF .PTR [SRC$B_SINKTYPE] EQLU .SNK  
: 1171    1159 3 |     THEN  
: 1172    1160 4 |         BEGIN  
: 1173    1161 4 |         .SRCPTR = .PTR;      ! Set source pointer for return  
: 1174    1162 4 |         STATUS = TRUE;  
: 1175    1163 4 |         EXITLOOP  
: 1176    1164 3 |         END;  
: 1177    1165 2 |     END;
```

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$GETNXTSNK Get next source block for specif

D 11
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 BLiss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 38
(13)

: 1178
: 1179
: 1180
: 1181
1166 2 RETURN .STATUS
1167 2
1168 2
1169 1 END:

! End of NML\$GETNXTSNK

				0004 00000	.ENTRY	NML\$GETNXTSNK, Save R2	1110
				0C 52 D4 00002	CLRL	STATUS	1153
				0C BC DD 00004	PUSHL	@SRCPTR	1154
				04 5E DD 00007	PUSHL	SP	1156
				04 AC DD 00009	PUSHL	BLKDSC	1
		00000000V	EF	02 FB 0000C	CALLS	#2, NML\$GETNXTSRC	1
		13		50 E9 00013	BLBC	R0, 2\$	1
		50		6E D0 00016	MOVL	PTR, R0	1158
08	AC	02	A0	00 ED 00019	CMPZV	#0, #8, 2(R0), SNK	1
				E5 12 00020	BNEQ	1\$	1
				50 D0 00022	MOVL	R0, @SRCPTR	1161
				01 D0 00026	MOVL	#1, STATUS	1162
				52 D0 00029	MOVL	STATUS, R0	1167
				04 0002C	RET		1169

: Routine Size: 45 bytes, Routine Base: \$CODE\$ + 0550

```
: 1183    1170 1 %SBTTL 'NML$GETNXTSRC Get next source block'  
: 1184    1171 1 GLOBAL ROUTINE NML$GETNXTSRC (BLKDSC, SRCPTR) =  
: 1185    1172 1  
: 1186    1173 1 ++  
: 1187    1174 1 FUNCTIONAL DESCRIPTION:  
: 1188    1175 1  
: 1189    1176 1 This routine searches the sink node buffer for the next source  
: 1190    1177 1 block.  
: 1191    1178 1  
: 1192    1179 1 FORMAL PARAMETERS:  
: 1193    1180 1  
: 1194    1181 1     BLKDSC      Descriptor of source block buffer.  
: 1195    1182 1     SRCPTR      Address of longword in which to return the address  
: 1196    1183 1           of the next source block. If value is within buffer  
: 1197    1184 1           range on input then it is used as the address of the  
: 1198    1185 1           starting source block.  
: 1199    1186 1  
: 1200    1187 1 IMPLICIT INPUTS:  
: 1201    1188 1  
: 1202    1189 1     NONE  
: 1203    1190 1  
: 1204    1191 1 IMPLICIT OUTPUTS:  
: 1205    1192 1  
: 1206    1193 1  
: 1207    1194 1  
: 1208    1195 1 ROUTINE VALUE:  
: 1209    1196 1 COMPLETION CODES:  
: 1210    1197 1  
: 1211    1198 1     TRUE is returned if a match is found, FALSE is returned if no match.  
: 1212    1199 1  
: 1213    1200 1 SIDE EFFECTS:  
: 1214    1201 1  
: 1215    1202 1  
: 1216    1203 1  
: 1217    1204 1 --  
: 1218    1205 1  
: 1219    1206 2 BEGIN  
: 1220    1207 2  
: 1221    1208 2 MAP  
: 1222    1209 2     BLKDSC : REF DESCRIPTOR;  
: 1223    1210 2  
: 1224    1211 2 LOCAL  
: 1225    1212 2     BUFEND,          ! Pointer to end of buffer  
: 1226    1213 2     PTR : REF BBLOCK,   ! Temporary source block pointer  
: 1227    1214 2     STATUS;        ! Routine status  
: 1228    1215 2  
: 1229    1216 2  
: 1230    1217 2 If descriptor indicates no source blocks (length=0) then  
: 1231    1218 2 return failure.  
: 1232    1219 2  
: 1233    1220 2 IF .BLKDSC [DSC$W_LENGTH] EQLU 0  
: 1234    1221 2 THEN  
: 1235    1222 2     RETURN FALSE;  
: 1236    1223 2  
: 1237    1224 2     BUFEND = .BLKDSC [DSC$A_POINTER] + .BLKDSC [DSC$W_LENGTH];  
: 1238    1225 2     PTR = ..SRCPTR;          ! Initialize source pointer  
: 1239    1226 2 !
```

```

: 1240      1227 2 | If PTR contains a value on input that is within the buffer range then
: 1241      1228 2 | use it as the starting point. If the value is not valid then return
: 1242      1229 2 | the address of the first source block in the buffer.
: 1243      1230 2 |
: 1244      1231 3 |   IF (.PTR LSSA .BLKDSC [DSC$A_POINTER])
: 1245      1232 2 |     OR
: 1246      1233 3 |     (.PTR GEQA .BUFEND)
: 1247      1234 2 |   THEN
: 1248      1235 2 |     PTR = .BLKDSC [DSC$A_POINTER]
: 1249      1236 2 |   ELSE
: 1250      1237 2 |     PTR = .PTR + .PTR [SRC$W_LENGTH];
: 1251      1238 2 |
: 1252      1239 2 | If pointer is still within range of buffer then return TRUE else
: 1253      1240 2 | return FALSE to indicate no more source blocks.
: 1254      1241 2 |
: 1255      1242 2 |   IF .PTR GEQA .BUFEND
: 1256      1243 2 |   THEN
: 1257      1244 2 |     STATUS = FALSE
: 1258      1245 2 |   ELSE
: 1259      1246 3 |     BEGIN
: 1260      1247 3 |       .SRCPTR = .PTR;           ! Set source pointer for return
: 1261      1248 3 |       STATUS = TRUE;
: 1262      1249 2 |     END;
: 1263      1250 2 |
: 1264      1251 2 | RETURN .STATUS
: 1265      1252 2 |
: 1266      1253 1 | END;                      ! End of NML$GETNXTSRC

```

				0004 00000	.ENTRY	NML\$GETNXTSRC, Save R2	
	51	04	AC	D0 00002	MOVL	BLKDSC, R1	1171
			61	B5 00006	TSTW	(R1)	1220
			2F	13 00008	BEQL	4\$	
	52	04	61	3C 0000A	MOVZWL	(R1), BUFEND	1224
	52	08	A1	C0 0000D	ADDL2	4(R1), BUFEND	
	50	08	BC	D0 00011	MOVL	@SRCPTR, PTR	1225
	04	A1	50	D1 00015	CMPL	PTR, 4(R1)	1231
			05	1F 00019	BLSSU	1\$	
	52		50	D1 0001B	CMPL	PTR, BUFEND	1233
			06	1F 0001E	BLSSU	2\$	
	50	04	A1	D0 00020	1\$: MOVL	4(R1), PTR	1235
			06	11 00024	BRB	3\$	
	51		60	3C 00026	2\$: MOVZWL	(PTR), R1	1237
	50		51	C0 00029	ADDL2	R1, PTR	
	52		50	D1 0002C	3\$: CMPL	PTR, BUFEND	1242
	08	BC	08	1E 0002F	BGEQU	4\$	
			50	D0 00031	MOVL	PTR, @SRCPTR	1247
		50	01	D0 00035	MOVL	#1, STATUS	1248
			04	00038	RET		1251
			50	D4 00039	4\$: CLRL	R0	1253
			04	0003B	RET		

; Routine Size: 60 bytes, Routine Base: \$CODE\$ + 057D

NML\$LOGOPS
V04-000

NML Logging data base operations module

NML\$GETNXTSRC Get next source block

G 11
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 41
(14)

NML
VO4

1268 1254 1 %SBTTL 'NML\$MATCHEV' Get event block matching specified class'
1269 1255 1 GLOBAL ROUTINE NML\$MATCHEV (SRCPTR, CLASS, EVTPTR) =
1270 1256 1
1271 1257 1 ++
1272 1258 1 FUNCTIONAL DESCRIPTION:
1273 1259 1
1274 1260 1 This routine searches the source block for an event block that
1275 1261 1 matches the specified class.
1276 1262 1
1277 1263 1 FORMAL PARAMETERS:
1278 1264 1
1279 1265 1 SRCPTR Pointer to source block.
1280 1266 1 CLASS Class code to match.
1281 1267 1 EVTPTR Address of longword in which the pointer to
1282 1268 1 the matched event block will be returned.
1283 1269 1
1284 1270 1 IMPLICIT INPUTS:
1285 1271 1 NONE
1286 1272 1
1287 1273 1 IMPLICIT OUTPUTS:
1288 1274 1 NONE
1289 1275 1
1290 1276 1
1291 1277 1
1292 1278 1 ROUTINE VALUE:
1293 1279 1 COMPLETION CODES:
1294 1280 1
1295 1281 1 TRUE is returned if a match is found, FALSE is returned if no match.
1296 1282 1
1297 1283 1 SIDE EFFECTS:
1298 1284 1
1299 1285 1 NONE
1300 1286 1
1301 1287 1 --
1302 1288 1
1303 1289 2 BEGIN
1304 1290 2
1305 1291 2 MAP
1306 1292 2 SRCPTR : REF BBLOCK;
1307 1293 2
1308 1294 2 LOCAL
1309 1295 2 PTR : REF BBLOCK, ! Temporary event block pointer
1310 1296 2 STATUS; ! Routine status
1311 1297 2
1312 1298 2 PTR = 0; ! Initialize source pointer
1313 1299 2 STATUS = FALSE; ! Initialize routine status
1314 1300 2
1315 1301 2 WHILE NML\$GETNXT_EVT (.SRCPTR, PTR) DO
1316 1302 3 BEGIN
1317 1303 3 IF .PTR [EVT\$W_CLASS] EQLU .CLASS
1318 1304 3 THEN
1319 1305 4 BEGIN
1320 1306 4 .EVTPTR = .PTR; ! Set event pointer for return
1321 1307 4 STATUS = TRUE;
1322 1308 4 EXITLOOP
1323 1309 3 END;
1324 1310 2 END;

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$MATCHEV~~T~~ Get event block matching specific

I 11

16-Sep-1984 00:19:25

14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 43
(15)

: 1325 1311 2 RETURN .STATUS
: 1326 1312 2
: 1327 1313 2
: 1328 1314 1 END;

! End of NML\$MATCHEV~~T~~

NML
V04

: 1
: 1
: 1
: 1

			0004 00000	.ENTRY NML\$MATCHEV T , Save R2	1255
			7E D4 00002	CLRL PTR	1298
			52 D4 00004	CLRL STATUS	1299
			5E DD 00006	PUSHL SP	1301
		04	1\$: AC DD 00008	PUSHL SRCPTR	
			02 FB 0000B	CALLS #2, NML\$GETNXTEVT	
			50 E9 00012	BLBC R0, 2\$	
08	AC	00 BE	10	CMPZV #0, #16, @PTR, CLASS	1303
			00 ED 00015	BNEQ 1\$	
		0C BC	E8 12 0001C	MOVL PTR, @EVTPTR	1306
			6E D0 0001E	MOVL #1, STATUS	1307
		52	01 D0 00022	MOVL STATUS, R0	1312
		50	52 D0 00025	2\$: RET	1314
			04 00028		:

: Routine Size: 41 bytes, Routine Base: \$CODE\$ + 05B9

```

: 1330      1315 1 %SBTTL 'NML$GETNXTEVT Get next event block'
: 1331      1316 1 GLOBAL ROUTINE NML$GETNXTEVT (SRCPTR, EVT PTR) =
: 1332      1317 1
: 1333      1318 1 ++
: 1334      1319 1 FUNCTIONAL DESCRIPTION:
: 1335      1320 1
: 1336      1321 1 This routine searches the source block for the next event block.
: 1337      1322 1
: 1338      1323 1 FORMAL PARAMETERS:
: 1339      1324 1
: 1340      1325 1 SRCPTR      Pointer to source block.
: 1341      1326 1 EVT PTR     Address of longword to contain address of matched
: 1342      1327 1 event block. If the value is within the event block
: 1343      1328 1 range then it is used as the starting event block
: 1344      1329 1 address.
: 1345      1330 1
: 1346      1331 1 IMPLICIT INPUTS:
: 1347      1332 1     NONE
: 1348      1333 1 IMPLICIT OUTPUTS:
: 1349      1334 1     NONE
: 1350      1335 1 ROUTINE VALUE:
: 1351      1336 1 COMPLETION CODES:
: 1352      1337 1
: 1353      1338 1
: 1354      1339 1     TRUE is returned if a match is found, FALSE is returned if no match.
: 1355      1340 1
: 1356      1341 1
: 1357      1342 1
: 1358      1343 1
: 1359      1344 1 SIDE EFFECTS:
: 1360      1345 1
: 1361      1346 1     NONE
: 1362      1347 1
: 1363      1348 1 --
: 1364      1349 1
: 1365      1350 2 BEGIN
: 1366      1351 2
: 1367      1352 2 MAP
: 1368      1353 2     SRCPTR : REF BBLOCK;
: 1369      1354 2
: 1370      1355 2 LOCAL
: 1371      1356 2     CLASSES,          | Number of event event blocks
: 1372      1357 2     MASKEND,         | Pointer to end of masks
: 1373      1358 2     MASKPTR,        | Pointer to masks
: 1374      1359 2     PTR : REF BBLOCK, | Temporary event block pointer
: 1375      1360 2     STATUS;          | Routine status
: 1376      1361 2
: 1377      1362 2     CLASSES = .SRCPTR [SRC$W_MSKCOUNT];
: 1378      1363 2
: 1379      1364 2     If no event masks are present (count=0) then
: 1380      1365 2     return failure.
: 1381      1366 2
: 1382      1367 2     IF .CLASSES EQLU 0
: 1383      1368 2     THEN
: 1384      1369 2     RETURN FALSE;
: 1385      1370 2
: 1386      1371 2     MASKPTR = .SRCPTR + SRC$K_LENGTH;

```

```

1387 1372 2 MASKEND = .MASKPTR + (.CLASSES * EVT$K_LENGTH);
1388 1373 2 PTR = ..EVTPTR; ! Initialize event pointer
1389 1374 2
1390 1375 2 If PTR contains a value on input that is within the buffer range then
1391 1376 2 use it as the starting point. If the value is not valid then return
1392 1377 2 the address of the first event block in the buffer.
1393 1378 2
1394 1379 3 IF (.PTR LSSA .MASKPTR)
1395 1380 2 OR
1396 1381 3 (.PTR GEQA .MASKEND)
1397 1382 2 THEN
1398 1383 2 PTR = .MASKPTR
1399 1384 2 ELSE
1400 1385 2 PTR = .PTR + EVT$K_LENGTH;
1401 1386 2
1402 1387 2 If pointer is still within range of buffer then return TRUE else
1403 1388 2 return FALSE to indicate no more event blocks.
1404 1389 2
1405 1390 2 IF .PTR GEQA .MASKEND
1406 1391 2 THEN
1407 1392 2 STATUS = FALSE
1408 1393 2 ELSE
1409 1394 3 BEGIN
1410 1395 3 .EVTPTR = .PTR; ! Set event pointer for return
1411 1396 3 STATUS = TRUE;
1412 1397 2 END;
1413 1398 2
1414 1399 2 RETURN .STATUS
1415 1400 2
1416 1401 1 END; ! End of NML$GETNXTEVT

```

			0004	00000	.ENTRY	NML\$GETNXT_EVT, Save R2	1316
50	04	AC	D0	00002	MOVL	SRCPTR, R0	1362
51	16	A0	3C	00006	MOVZWL	22(R0), CLASSES	
		2D	13	0000A	BEQL	4S	1367
50		18	C0	0000C	ADDL2	#24, MASKPTR	1371
51		14	C4	0000F	MULL2	#20, R1	1372
51	08	50	C1	00012	ADDL3	MASKPTR, R1, MASKEND	
51		BC	D0	00016	MOVL	AEVTPTR, PTR	1373
50		51	D1	0001A	CMPL	PTR, MASKPTR	1379
52		05	1F	0001D	BLSSU	1S	
51		51	D1	0001F	CMPL	PTR, MASKEND	1381
		05	1F	00022	BLSSU	2S	
51		50	D0	00024	1\$: MOVL	MASKPTR, PTR	1383
		03	11	00027	BRB	3S	
51		14	C0	00029	2\$: ADDL2	#20, PTR	1385
52		51	D1	0002C	3\$: CMPL	PTR, MASKEND	1390
08	BC	08	1E	0002F	BGEQU	4S	
		51	D0	00031	MOVL	PTR, AEVTPTR	1395
50		01	D0	00035	MOVL	#1, STATUS	1396
		04	00038		RET		1399
50		50	D4	00039	4\$: CLRL	R0	
		04	0003B		RET		1401

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$GETNXT_EVT Get next event block

L 11
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 46
(16)

; Routine Size: 60 bytes. Routine Base: \$CODE\$ + 05E2

```
: 1418      1402 1 %SBTTL 'NML$BLDSRC Build a source block'  
1419      1403 1 GLOBAL ROUTINE NML$BLDSRC (BUFDSC, SNK, SRC, ENTDSC) : NOVALUE =  
1420      1404 1  
1421      1405 1 ++  
1422      1406 1 FUNCTIONAL DESCRIPTION:  
1423      1407 1  
1424      1408 1 This routine builds a source block.  
1425      1409 1  
1426      1410 1 FORMAL PARAMETERS:  
1427      1411 1  
1428      1412 1 BUFDSC Descriptor of buffer to hold new source block.  
1429      1413 1 (Assumed to be at least SRC$K_LENGTH bytes.)  
1430      1414 1 SNK Logging sink type code.  
1431      1415 1 SRC Event source type code.  
1432      1416 1 ENTDSC Event source id string descriptor.  
1433      1417 1  
1434      1418 1 IMPLICIT INPUTS:  
1435      1419 1  
1436      1420 1 NONE  
1437      1421 1  
1438      1422 1 IMPLICIT OUTPUTS:  
1439      1423 1  
1440      1424 1 NONE  
1441      1425 1  
1442      1426 1 ROUTINE VALUE:  
1443      1427 1 COMPLETION CODES:  
1444      1428 1  
1445      1429 1 NONE  
1446      1430 1  
1447      1431 1 SIDE EFFECTS:  
1448      1432 1  
1449      1433 1 NONE  
1450      1434 1  
1451      1435 1 --  
1452      1436 1  
1453      1437 2 BEGIN  
1454      1438 2  
1455      1439 2 MAP  
1456      1440 2 BUFDSC : REF DESCRIPTOR,  
1457      1441 2 ENTDSC : REF DESCRIPTOR;  
1458      1442 2  
1459      1443 2 LOCAL  
1460      1444 2 SRCPTR : REF BBLOCK;  
1461      1445 2  
1462      1446 2 SRCPTR = .BUFDSC [DSC$A_POINTER];  
1463      1447 2 CH$FILL (0, SRC$K_LENGTH, .SRCPTR); ! Zero the event block  
1464      1448 2  
1465      1449 2 SRCPTR [SRC$W_LENGTH] = SRC$K_LENGTH;  
1466      1450 2 SRCPTR [SRC$B_SINKTYPE] = .SNK;  
1467      1451 2 SRCPTR [SRC$B_SRCTYPE] = .SRC;  
1468      1452 2  
1469      1453 2 SELECTONEU .SRC OF  
1470      1454 2 SET  
1471      1455 2 [NMASC_ENT_NOD]: ! Node  
1472      1456 2  
1473      1457 2 CH$MOVE {2,  
1474      1458 2 ENTDSC [DSC$A_POINTER],
```

NMLS LOGO OPS
V04-000

NML Logging data base operations module NML\$BLDSRC Build a source block

N 11
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32:1

Page 48
(17)

```

1475      1459  2      SRCPTR [SRC$W_NODADR]);
1476      1460  2
1477      1461  2      [NMASC_ENT_CIR,
1478      1462  2      NMASC_ENT_LIN,
1479      1463  2      NMASC_ENT_MOD]:           ! Circuit or Line or Module
1480      1464  2      BEGIN
1481      1465  3
1482      1466  3      SRCPTR [SRC$B_IDLENGTH] = .ENTDSC [DSC$W_LENGTH];
1483      1467  3      CHSMOVE (.ENTDSC [DSC$W_LENGTH],
1484      1468  3          .ENTDSC [DSC$A_POINTER],
1485      1469  3          SRCPTR [SRC$T_ID]);
1486      1470  3
1487      1471  2      END;
1488      1472  2
1489      1473  2      TES;
1490      1474  2
1491      1475  1      END;                           ! End of NML$BLDSRC

```

; Routine Size: 76 bytes, Routine Base: \$CODE\$ + 061E

```
: 1493 1476 1 %SBTTL 'NML$BLDEVT Build an event class block'  
1494 1477 1 GLOBAL ROUTINE NML$BLDEVT (FCT, CLASS, MSKLEN, MSKPTR, EVT PTR) : NOVALUE =  
1495 1478 1  
1496 1479 1 ++  
1497 1480 1 | FUNCTIONAL DESCRIPTION:  
1498 1481 1 | This routine builds an event class block.  
1499 1482 1  
1500 1483 1 | FORMAL PARAMETERS:  
1501 1484 1  
1502 1485 1  
1503 1486 1 | FCT Mask operation code. (0=CLEAR, 1=SET)  
1504 1487 1 | CLASS Event class code.  
1505 1488 1 | MSKLEN Length in bytes of event mask.  
1506 1489 1 | MSKPTR Address of event mask.  
1507 1490 1 | EVT PTR Address of event block to be filled in.  
1508 1491 1  
1509 1492 1 | IMPLICIT INPUTS:  
1510 1493 1  
1511 1494 1 | NONE  
1512 1495 1  
1513 1496 1 | IMPLICIT OUTPUTS:  
1514 1497 1  
1515 1498 1 | NONE  
1516 1499 1  
1517 1500 1 | ROUTINE VALUE:  
1518 1501 1 | COMPLETION CODES:  
1519 1502 1  
1520 1503 1 | NONE  
1521 1504 1  
1522 1505 1 | SIDE EFFECTS:  
1523 1506 1  
1524 1507 1 | NONE  
1525 1508 1  
1526 1509 1 --  
1527 1510 1  
1528 1511 2 | BEGIN  
1529 1512 2  
1530 1513 2 | MAP  
1531 1514 2 | EVT PTR : REF BBLOCK;  
1532 1515 2  
1533 1516 2 | CH$FILL (0, EVT$K_LENGTH, .EVT PTR); ! Zero the event block  
1534 1517 2  
1535 1518 2 | EVT PTR [EVT$W_CLASS] = .CLASS; ! Fill in the class code  
1536 1519 2  
1537 1520 2 | If function is SET (FCT=1) then move the mask into the log mask.  
1538 1521 2 | Otherwise (FCT=0), function is CLEAR so move the mask into the filter  
1539 1522 2 | mask.  
1540 1523 2  
1541 1524 2 | IF .FCT  
1542 1525 2 | THEN  
1543 1526 2 | | CH$MOVE (.MSKLEN, .MSKPTR, EVT PTR [EVT$Q_LOGMSK])  
1544 1527 2 | ELSE  
1545 1528 2 | | CH$MOVE (.MSKLEN, .MSKPTR, EVT PTR [EVT$Q_FILTERMSK]);  
1546 1529 2  
1547 1530 1 | END; ! End of NML$BLDEVT
```

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$BLDEVT Build an event class block

C 12
16-Sep-1984 00:19:25 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:50:11 [NML.SRC]NMLLOGOPS.B32;1

Page 50
(18)

NML:
VO4

19

: R

							007C 00000	.ENTRY	NML\$BLDEVT, Save R2,R3,R4,R5,R6	: 1477
14		00	56	14	AC	D0	00002	MOVL	EVT PTR, R6	: 1516
			6E		00	2C	00006	MOVCS	#0, (SP), #0, #20, (R6)	
					66		0000B			
			66	08	AC	B0	0000C	MOVW	CLASS, (R6)	: 1518
			08	04	AC	F9	00010	BLBC	FCT, fs	: 1524
04	A6	10	BC	0C	AC	28	00014	MOVCS	MSKLEN, @MSKPTR, 4(R6)	: 1526
					04		0001B	RET		
0C	A6	10	BC	0C	AC	28	0001C	1\$: MOVCS	MSKLEN, @MSKPTR, 12(R6)	: 1528
					04		00023	RET		: 1530

; Routine Size: 36 bytes, Routine Base: \$CODE\$ + 066A

```

1549      1531 1 XSBTTL 'NML$ADDSSRC Add a source block to buffer'
1550      1532 1 GLOBAL ROUTINE NML$ADDSSRC (BUFDSC, SRCDSC, SRCPTR) =
1551      1533 1
1552      1534 1 ++
1553      1535 1 FUNCTIONAL DESCRIPTION:
1554      1536 1
1555      1537 1 This routine adds a source block to the specified buffer.
1556      1538 1
1557      1539 1 FORMAL PARAMETERS:
1558      1540 1
1559      1541 1     BUFDSC      Descriptor of source block buffer.
1560      1542 1     SRCDSC      Descriptor of source block data in buffer.
1561      1543 1     SRCPTR      Pointer to source block to be added.
1562      1544 1
1563      1545 1 IMPLICIT INPUTS:
1564      1546 1
1565      1547 1     NONE
1566      1548 1
1567      1549 1 IMPLICIT OUTPUTS:
1568      1550 1
1569      1551 1     NONE
1570      1552 1
1571      1553 1 ROUTINE VALUE:
1572      1554 1 COMPLETION CODES:
1573      1555 1
1574      1556 1     Returns TRUE if the source block was added. Returns FALSE if
1575      1557 1     there was not enough room in the buffer.
1576      1558 1
1577      1559 1 SIDE EFFECTS:
1578      1560 1
1579      1561 1     NONE
1580      1562 1
1581      1563 1 --
1582      1564 1
1583      1565 2 BEGIN
1584      1566 2
1585      1567 2 MAP
1586      1568 2     BUFDSC : REF DESCRIPTOR,
1587      1569 2     SRCDSC : REF DESCRIPTOR,
1588      1570 2     SRCPTR : REF BBLOCK;
1589      1571 2
1590      1572 2
1591      1573 2     Make sure source block will fit in the buffer.
1592      1574 2
1593      1575 3     IF (.BUFDSC [DSC$W_LENGTH] - .SRCDSC [DSC$W_LENGTH])
1594      1576 2         LSS
1595      1577 2         .SRCPTR [SRC$W_LENGTH]
1596      1578 2     THEN
1597      1579 2         RETURN FALSE;
1598      1580 2
1599      1581 2     Block will fit so move it.
1600      1582 2
1601      1583 2     CH$MOVE (.SRCPTR [SRC$W_LENGTH],
1602      1584 2             .SRCPTR,
1603      1585 2             .SRCDSC [DSC$A_POINTER] + .SRCDSC [DSC$W_LENGTH]);
1604      1586 2
1605      1587 2     Calculate resulting buffer length.

```

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$ADDSRC Add a source block to buffer

E 12
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11
VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 52
(19)

: 1606 1588 2 !
: 1607 1589 2 SRCDSC [DSC\$W_LENGTH] =
: 1608 1590 2 .SRCDSC [DSC\$W_LENGTH] + .SRCPTR [SRC\$W_LENGTH];
: 1609 1591 2
: 1610 1592 2 RETURN TRUE
: 1611 1593 2
: 1612 1594 1 END:
! End of NML\$ADDSRC

				007C 00000	.ENTRY	NML\$ADDSRC, Save R2,R3,R4,R5,R6	:	1532	
			56	08	AC	DO 00002	MOVL	SRCDSC, R6	
			50	04	BC	3C 00006	MOVZWL	@BUFDS, R0	
			51		66	3C 0000A	MOVZWL	(R6), R1	
			50		51	C2 0000D	SUBL2	R1, R0	
50	OC BC		10		00	ED 00010	CMPZV	#0, #16, @SRCPTR, R0	
					15	14 00016	BGTR	1\$	1577
			50		66	3C 00018	MOVZWL	(R6), R0	
			50	04	A6	C0 0001B	ADDL2	4(R6), R0	1585
			60	OC BC	0C	BC 28 0001F	MOVC3	@SRCPTR, @SRCPTR, (R0)	
			66	0C	BC	A0 00025	ADDW2	@SRCPTR, (R6)	
			50		01	D0 00029	MOVL	#1, R0	
					04	0002C	RET		1590
					50	D4 0002D 1\$:	CLRL	R0	
					04	0002F	RET		1592
									1594

: Routine Size: 48 bytes, Routine Base: \$CODE\$ + 068E

**F

```
1614    1595 1 %SBTTL 'NML$REPSRC Replace a source block in buffer'
1615    1596 1 GLOBAL ROUTINE NML$REPSRC (BUFDSC, SRCDSC, OLDSRC, NEWSRC) =
1616    1597 1
1617    1598 1 ++
1618    1599 1 FUNCTIONAL DESCRIPTION:
1619    1600 1
1620    1601 1 This routine adds a source block to the specified buffer.
1621    1602 1
1622    1603 1 FORMAL PARAMETERS:
1623    1604 1
1624    1605 1     BUFDSC      Descriptor of source block buffer.
1625    1606 1     SRCDSC      Descriptor of source block data in buffer.
1626    1607 1     OLDSRC      Pointer to old source block in buffer.
1627    1608 1     NEWSRC      Pointer to source block to be added.
1628    1609 1
1629    1610 1 IMPLICIT INPUTS:
1630    1611 1
1631    1612 1     NONE
1632    1613 1
1633    1614 1 IMPLICIT OUTPUTS:
1634    1615 1
1635    1616 1     NONE
1636    1617 1
1637    1618 1 ROUTINE VALUE:
1638    1619 1 COMPLETION CODES:
1639    1620 1
1640    1621 1     Returns TRUE if the source block was added. Returns FALSE if
1641    1622 1     there was not enough room in the buffer.
1642    1623 1
1643    1624 1 SIDE EFFECTS:
1644    1625 1
1645    1626 1     NONE
1646    1627 1
1647    1628 1 --
1648    1629 1
1649    1630 2 BEGIN
1650    1631 2
1651    1632 2 MAP
1652    1633 2     BUFDSC : REF DESCRIPTOR,
1653    1634 2     SRCDSC : REF DESCRIPTOR,
1654    1635 2     OLDSRC : REF BBLOCK,
1655    1636 2     NEWSRC : REF BBLOCK;
1656    1637 2
1657    1638 2 LOCAL
1658    1639 2     FREELEN,
1659    1640 2     MOVLEN;
1660    1641 2
1661    1642 2     Make sure source block will fit in the buffer.
1662    1643 2
1663    1644 2     FREELEN = .BUFDSC [DSCSW_LENGTH] -
1664    1645 2             .SRCDSC [DSCSW_LENGTH] +
1665    1646 2             .OLDSRC [SRC$W_LENGTH];
1666    1647 2     IF .FREELEN LSS .NEWSRC [SRC$W_LENGTH]
1667    1648 2     THEN
1668    1649 2         RETURN FALSE;
1669    1650 2
1670    1651 2     FREELEN = .FREELEN - .NEWSRC [SRC$W_LENGTH];
```

```

1671      1652 2 1
1672      1653 2 1 Block will fit so move it.
1673
1674      1655 2 1 MOVLEN = .SRCDSR [DSC$A_POINTER] + .SRCDSR [DSC$W_LENGTH];
1675      1656 2 1 MOVLEN = .MOVLEN - .OLDSRC;
1676      1657 2 1 MOVLEN = .MOVLEN - .OLDSRC [SRC$W_LENGTH];
1677      1658 2
1678      1659 2 1 CH$MOVE (.MOVLEN,
1679      1660 2 1     :OLDSRC + .OLDSRC [SRC$W_LENGTH],
1680      1661 2 1     :OLDSRC + .NEWSRC [SRC$W_LENGTH]);
1681      1662 2
1682      1663 2 1 CH$MOVE (.NEWSRC [SRC$W_LENGTH],
1683      1664 2 1     .NEWSRC,
1684      1665 2 1     .OLDSRC);
1685
1686      1666 2 1 Calculate resulting buffer length.
1687      1667 2 1 SRCDSR [DSC$W_LENGTH] =
1688      1669 2 1     .BUFDSC [DSC$W_LENGTH] - .FREELEN;
1689      1670 2
1690      1671 2 1 RETURN TRUE
1691      1672 2
1692      1673 2
1693      1674 1 1 END:

```

! End of NML\$ADDSRC

			03FC 00000	.ENTRY	NML\$REPSRC, Save R2,R3,R4,R5,R6,R7,R8,R9	1596
	58	08	AC D0 00002	MOVL	SRCDSR, R8	1645
	50	04	BC 3C 00006	MOVZWL	@BUFDSC, R0	
	51	68	3C 0000A	MOVZWL	(R8), R1	
	50	51	C2 0000D	SUBL2	R1, R0	
	56	0C	AC D0 00010	MOVL	OLDSRC, R6	1646
	51	66	3C 00014	MOVZWL	(R6), R1	
	50	51	C1 00017	ADDL3	R1, R0, FREELEN	
59	57	10	BC 3C 0001B	MOVZWL	@NEWSRC, R7	1647
	57	59	D1 0001F	CMPL	FREELEN, R7	
		24	19 00022	BLSS	1\$	
	59	57	C2 00024	SUBL2	R7, FREELEN	1651
	50	68	3C 00027	MOVZWL	(R8), MOVLEN	1655
	50	04	A8 C0 0002A	ADDL2	4(R8), MOVLEN	
	50	56	C2 0002E	SUBL2	R6, MOVLEN	1656
	50	51	C2 00031	SUBL2	R1, MOVLEN	1657
6746	6146	50	28 00034	MOVC3	MOVLEN, (R1)[R6], (R7)[R6]	1661
66	10	BC	57 28 0003A	MOVC3	R7, @NEWSRC, (R6)	1665
68	04	BC	59 A3 0003F	SUBW3	FREELEN, @BUFDSC, (R8)	1670
	50	01	D0 00044	MOVL	#1, R0	1672
		04	00047	RET		
		50	D4 00048 1\$:	CLRL	R0	1674
		04	0004A	RET		

: Routine Size: 75 bytes, Routine Base: \$CODE\$ + 06BE

: 1695 1675 1 %SBTTL 'NML\$REMSRC Remove source block from buffer'
1696 1676 1 GLOBAL ROUTINE NML\$REMSRC (BLKDSC, SRCPTR) : NOVALUE =
1697 1677 1
1698 1678 1 !++
1699 1679 1 FUNCTIONAL DESCRIPTION:
1700 1680 1
1701 1681 1 This routine removes the specified source block from the buffer.
1702 1682 1
1703 1683 1 FORMAL PARAMETERS:
1704 1684 1
1705 1685 1 BLKDSC Descriptor of source block buffer.
1706 1686 1 SRCPTR Pointer to source block in buffer to be removed.
1707 1687 1
1708 1688 1 IMPLICIT INPUTS:
1709 1689 1
1710 1690 1 NONE
1711 1691 1
1712 1692 1 IMPLICIT OUTPUTS:
1713 1693 1
1714 1694 1 NONE
1715 1695 1
1716 1696 1 ROUTINE VALUE:
1717 1697 1 COMPLETION CODES:
1718 1698 1
1719 1699 1 NONE
1720 1700 1
1721 1701 1 SIDE EFFECTS:
1722 1702 1
1723 1703 1 NONE
1724 1704 1
1725 1705 1 --
1726 1706 1
1727 1707 2 BEGIN
1728 1708 2
1729 1709 2 MAP
1730 1710 2 BLKDSC : REF DESCRIPTOR,
1731 1711 2 SRCPTR : REF BBLOCK;
1732 1712 2
1733 1713 2 LOCAL
1734 1714 2 BUFEND,
1735 1715 2 LEN,
1736 1716 2 PTR;
1737 1717 2
1738 1718 2 Set up length and pointers to remove source block.
1739 1719 2
1740 1720 2 LEN = .SRCPTR [SRCSW_LENGTH];
1741 1721 2 PTR = .SRCPTR + .LEN;
1742 1722 2 BUFEND = .BLKDSC [DSCSA_POINTER] + .BLKDSC [DSCSW_LENGTH];
1743 1723 2
1744 1724 2 Move the end of the buffer back over the source block to be removed.
1745 1725 2
1746 1726 2 CH\$MOVE (.BUFEND - .PTR,
1747 1727 2 .PTR,
1748 1728 2 .SRCPTR);
1749 1729 2
1750 1730 2 Update the descriptor.
1751 1731 2

NML\$LOGOPS
VO4-000

NML Logging data base operations module
NML\$REMSRC Remove source block from buffer

i 12

16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 56
(21)

: 1752 1732 2 BLKDSC [DSC\$W_LENGTH] =
: 1753 1733 2 .BLKDSC [DSC\$W_LENGTH] - .LEN;
: 1754 1734 2
: 1755 1735 1 END;

! End of NML\$REMSRC

			00FC 00000	.ENTRY NML\$REMSRC, Save R2,R3,R4,R5,R6,R7	1676
		51	57 08 BC 3C 00002	MOVZWL @SRCPTR, LEN	1720
			57 08 AC C1 00006	ADDL3 SRCPTR, LEN, PTR	1721
			56 04 AC D0 0000B	MOVL BLKDSC, R6	1722
			50 66 3C 000CF	MOVZWL (R6), BUFEND	
			50 04 A6 C0 00012	ADDL2 4(R6), BUFEND	
			50 51 C2 00016	SUBL2 PTR, R0	1726
08	BC		61 50 28 00019	MOVC3 R0, (PTR), @SRCPTR	1728
			66 57 A2 0001E	SUBW2 LEN, (R6)	1733
			04 00021	RET	1735

; Routine Size: 34 bytes, Routine Base: \$CODE\$ + 0709

```
: 1757    1736 1 %SBTTL 'NML$ADDEVT Add an event block to source buffer'  
: 1758    1737 1 GLOBAL ROUTINE NML$ADDEVT (BUFDSC, SRCPTR, EVTPTR) =  
: 1759    1738 1  
: 1760    1739 1 ++  
: 1761    1740 1 | FUNCTIONAL DESCRIPTION:  
: 1762    1741 1 |  
: 1763    1742 1 | This routine adds an event block to the specified source buffer.  
: 1764    1743 1  
: 1765    1744 1 | FORMAL PARAMETERS:  
: 1766    1745 1  
: 1767    1746 1 | BUFDSC      Descriptor of buffer containing source block.  
: 1768    1747 1 | SRCPTR      Pointer to source block in buffer.  
: 1769    1748 1 | EVTPTR      Pointer to event block to be added.  
: 1770    1749 1  
: 1771    1750 1 | IMPLICIT INPUTS:  
: 1772    1751 1 |  
: 1773    1752 1 |     NONE  
: 1774    1753 1  
: 1775    1754 1 | IMPLICIT OUTPUTS:  
: 1776    1755 1  
: 1777    1756 1 |  
: 1778    1757 1 |     NONE  
: 1779    1758 1 | ROUTINE VALUE:  
: 1780    1759 1 | COMPLETION CODES:  
: 1781    1760 1  
: 1782    1761 1 | Returns TRUE if the event block was added. Returns FALSE if  
: 1783    1762 1 | there was not enough room in the buffer.  
: 1784    1763 1  
: 1785    1764 1 | SIDE EFFECTS:  
: 1786    1765 1  
: 1787    1766 1 |  
: 1788    1767 1 |     NONE  
: 1789    1768 1 |--  
: 1790    1769 1  
: 1791    1770 2 | BEGIN  
: 1792    1771 2  
: 1793    1772 2 | MAP  
: 1794    1773 2 |     BUFDSC : REF DESCRIPTOR,  
: 1795    1774 2 |     SRCPTR : REF BBLOCK,  
: 1796    1775 2 |     EVTPTR : REF BBLOCK;  
: 1797    1776 2  
: 1798    1777 2 | Make sure event block will fit in the buffer.  
: 1799    1778 2  
: 1800    1779 3 | IF (.BUFDSC [DSC$W_LENGTH] - .SRCPTR [SRC$W_LENGTH])  
: 1801    1780 2 |     LSS  
: 1802    1781 2 |     EVT$K_LENGTH  
: 1803    1782 2 | THEN  
: 1804    1783 2 |     RETURN FALSE;  
: 1805    1784 2  
: 1806    1785 2 | Block will fit so move it.  
: 1807    1786 2  
: 1808    1787 2 |     CH$MOVE (EVT$K_LENGTH,  
: 1809    1788 2 |             .EVTPTR,  
: 1810    1789 2 |             .SRCPTR + .SRCPTR [SRC$W_LENGTH]);  
: 1811    1790 2  
: 1812    1791 2 | Calculate resulting buffer length and store it in source block.  
: 1813    1792 2 | Also increment the mask count.
```

```

: 1814    1793 2 !
: 1815    1794 2 SRCPTR [SRC$W_LENGTH] =
: 1816    1795 2     .SRCPTR [SRC$W_LENGTH] + EVTSK_LENGTH;
: 1817    1796 2
: 1818    1797 2 SRCPTR [SRC$W_MSKCOUNT] =
: 1819    1798 2     .SRCPTR [SRC$W_MSKCOUNT] + 1;
: 1820    1799 2
: 1821    1800 2 RETURN TRUE
: 1822    1801 2
: 1823    1802 1 END;                                ! End of NML$ADDEVT

```

				007C 00000	.ENTRY	NML\$ADDEVT, Save R2,R3,R4,R5,R6	:	1737
			56	08 AC D0 00002	MOVL	SRCPTR, R6	:	1779
			51	66 3C 00006	MOVZWL	(R6), R1	:	
			50	14 A1 9E 00009	MOVAB	20(R1), R0	:	1780
50	04 BC		10	00 ED 0000D	CMPZV	#0, #16, @BUFDSC, R0	:	
				10 19 00013	BLSS	1\$:	
		6146	0C BC	14 28 00015	MOVC3	#20, @EVTPTR, (R1)[R6]	:	1789
			66	14 A0 0001B	ADDW2	#20, (R6)	:	1795
				16 A6 B6 0001E	INCW	22(R6)	:	1798
			50	01 D0 00021	MOVL	#1, R0	:	1800
				04 00024	RET		:	
				50 D4 00025 1\$:	CLRL	R0	:	1802
				04 00027	RET		:	

: Routine Size: 40 bytes, Routine Base: \$CODE\$ + 072B

1825 1803 1 %SBTTL 'NML\$MODEVT Modify event block'
1826 1804 1 GLOBAL ROUTINE NML\$MODEVT (FCT, ZER, EVT PTR, MSKLEN, MSKPTR) : NOVALUE =
1827 1805 1
1828 1806 1 ++
1829 1807 1 FUNCTIONAL DESCRIPTION:
1830 1808 1 This routine the modifies the specified event block.
1831 1809 1
1832 1810 1 FORMAL PARAMETERS:
1833 1811 1
1834 1812 1
1835 1813 1 FCT Mask operation code. (FALSE=CLEAR, TRUE=SET).
1836 1814 1 ZER Zero flag. (TRUE=yes, FALSE=no).
1837 1815 1 EVT PTR Pointer to event block.
1838 1816 1 MSKLEN Length of mask value to be added.
1839 1817 1 MSKPTR Pointer to mask value to be added.
1840 1818 1
1841 1819 1 IMPLICIT INPUTS:
1842 1820 1 NONE
1843 1821 1
1844 1822 1 IMPLICIT OUTPUTS:
1845 1823 1 NONE
1846 1824 1
1847 1825 1
1848 1826 1
1849 1827 1 ROUTINE VALUE:
1850 1828 1 COMPLETION CODES:
1851 1829 1
1852 1830 1 NONE
1853 1831 1
1854 1832 1 SIDE EFFECTS:
1855 1833 1
1856 1834 1 NONE
1857 1835 1
1858 1836 1 --
1859 1837 1
1860 1838 2 BEGIN
1861 1839 2
1862 1840 2 MAP
1863 1841 2 EVT PTR : REF BBLOCK,
1864 1842 2 MSK PTR : REF BITVECTOR;
1865 1843 2
1866 1844 2 LOCAL
1867 1845 2 BITLEN, | Length of mask in bits
1868 1846 2 OLDMASK : REF BITVECTOR, | Mask not changed
1869 1847 2 RESMSK : REF BITVECTOR; | Address of result mask
1870 1848 2
1871 1849 2 If the operation is SET (FCT=1) then modify log mask.
1872 1850 2 Otherwise, operation is CLEAR (FCT=0) so modify filter mask.
1873 1851 2
1874 1852 2 IF .FCT
1875 1853 2 THEN
1876 1854 3 BEGIN
1877 1855 3 RESMSK = EVT PTR [EVT\$Q_LOGMSK];
1878 1856 3 OLDMASK = EVT PTR [EVT\$Q_FILTERMSK];
1879 1857 3 END
1880 1858 2 ELSE
1881 1859 3 BEGIN

```

1882    1860 3      RESMSK = EVT PTR [EVT$Q_FILTERMSK];
1883    1861 3      OLDMASK = EVT PTR [EVT$Q_LOGMSK];
1884    1862 2      END;
1885    1863 2      :
1886    1864 2      Set the correct bits in the result mask.
1887    1865 2      :
1888    1866 2      BITLEN = .MSKLEN * 8;
1889    1867 2      :
1890    1868 2      INCR I FROM 0 TO .BITLEN - 1 DO
1891    1869 3      BEGIN
1892    1870 3      :
1893    1871 3      RESMSK [.I] = :RESMSK [.I] OR .MSKPTR [.I];
1894    1872 3      OLDMASK [.I] = :OLDMASK [.I] AND NOT .MSKPTR [.I];
1895    1873 3      :
1896    1874 2      END;
1897    1875 2      :
1898    1876 2      If the other mask should be zeroed (ZER=TRUE) then zero it.
1899    1877 2      :
1900    1878 2      IF .ZER
1901    1879 2      THEN
1902    1880 3      BEGIN
1903    1881 3      :
1904    1882 3      MAP OLDMASK : REF VECTOR [, BYTE];
1905    1883 3      :
1906    1884 3      INCR I FROM 0 TO EVT$S_LOGMSK - 1 DO
1907    1885 4      BEGIN
1908    1886 4      :
1909    1887 4      OLDMASK [.I] = 0;
1910    1888 4      :
1911    1889 3      END;
1912    1890 2      END;
1913    1891 2      END;
1914    1892 1      END;

```

! End of NML\$MODEVT

					003C 00000	.ENTRY	NML\$MODEVT, Save R2,R3,R4,R5	1804
51	50	OC	AC	04	04 C1 00002	ADDL3	#4, EVT PTR, R1	1855
		OC	AC	05	0C C1 00007	ADDL3	#12, EVT PTR, R0	1856
				53	AC E9 0000C	BLBC	FCT, 1\$	1852
				54	50 7D 00010	MOVQ	R0, OLDMASK	1856
				53	06 11 00013	BRB	2\$	1852
				54	50 D0 00015	1\$: MOVL	R0, RESMSK	1860
				53	51 D0 00018	MOVL	R1, OLDMASK	1861
		55	10	AC	03 78 0001B	2\$: ASHL	#3, MSKLEN, BITLEN	1866
				51	01 CE 00020	MNEGL	#1, I	1872
					26 11 00023	BRB	4\$	
52	50	64	01	51	51 EF 00025	3\$: EXTZV	I, #1, (RESMSK), R2	1871
	14	BC	01	50	51 EF 0002A	EXTZV	I, #1, @MSKPTR, R0	
64		01	51	52	52 C8 00030	BISL2	R2, R0	
52		63	01	50	50 F0 00033	INSV	R0, I, #1, (RESMSK)	
50	14	BC	01	51	51 EF 00038	EXTZV	I, #1, (OLDMASK), R2	1872
				52	51 EF 0003D	EXTZV	I, #1, @MSKPTR, R0	
63		01	51	50	50 CA 00043	BICL2	R0, R2	
				52	52 F0 00046	INSV	R2, I, #1, (OLDMASK)	

NML SLOGOPS
V04-000

NML Logging data base operations module
NML\$MODEVT Modify event block

N 12
16-527

16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 61
(23)

NM
VO

D6	51		55	F2	0004B	4\$:	A0BLSS	BITLEN	I, 3\$
	09	08	AC	E9	0004F		BLBC	ZER,	6\$
			51	D4	00053		CLRL	I	
F9	51		6143	94	00055	5\$:	CLR8	(1)[OLDMSK]	
			07	F3	00058		A0BLEQ	#7, I,	5\$
				04	0005C	6\$:	RET		

; Routine Size: 93 bytes, Routine Base: \$CODE\$ + 0753

```
: 1916    1893 1 %SBTTL 'NML$REMEVT Remove event block from source buffer'  
: 1917    1894 1 GLOBAL ROUTINE NML$REMEVT (SRCPTR, EVT PTR) : NOVALUE =  
: 1918    1895 1  
: 1919    1896 1 ++  
: 1920    1897 1 FUNCTIONAL DESCRIPTION:  
: 1921    1898 1  
: 1922    1899 1 This routine removes the specified event block from the source buffer.  
: 1923    1900 1  
: 1924    1901 1 FORMAL PARAMETERS:  
: 1925    1902 1  
: 1926    1903 1 SRCPTR      Pointer to source block.  
: 1927    1904 1 EVT PTR     Pointer to event block to be removed from source.  
: 1928    1905 1  
: 1929    1906 1 IMPLICIT INPUTS:  
: 1930    1907 1  
: 1931    1908 1 NONE  
: 1932    1909 1  
: 1933    1910 1 IMPLICIT OUTPUTS:  
: 1934    1911 1  
: 1935    1912 1 NONE  
: 1936    1913 1  
: 1937    1914 1 ROUTINE VALUE:  
: 1938    1915 1 COMPLETION CODES:  
: 1939    1916 1  
: 1940    1917 1 NONE  
: 1941    1918 1  
: 1942    1919 1 SIDE EFFECTS:  
: 1943    1920 1  
: 1944    1921 1 NONE  
: 1945    1922 1  
: 1946    1923 1 --  
: 1947    1924 1  
: 1948    1925 2 BEGIN  
: 1949    1926 2  
: 1950    1927 2 MAP  
: 1951    1928 2 SRCPTR : REF BBLOCK,  
: 1952    1929 2 EVT PTR : REF BBLOCK;  
: 1953    1930 2  
: 1954    1931 2 LOCAL  
: 1955    1932 2 BUFEND,  
: 1956    1933 2 PTR;  
: 1957    1934 2  
: 1958    1935 2 Set up length and pointers to remove event block.  
: 1959    1936 2  
: 1960    1937 2 PTR = .EVT PTR + EVT$K LENGTH;  
: 1961    1938 2 BUFEND = .SRC PTR + .SRC PTR [SRC$W_LENGTH];  
: 1962    1939 2  
: 1963    1940 2 Move the end of the buffer back over the event block to be removed.  
: 1964    1941 2  
: 1965    1942 2 CH$MOVE (.BUFEND - .PTR,  
: 1966    1943 2 .PTR,  
: 1967    1944 2 .EVT PTR);  
: 1968    1945 2  
: 1969    1946 2 Update the length of the source block.  
: 1970    1947 2 Also decrement the mask count.  
: 1971    1948 2  
: 1972    1949 2 SRC PTR [SRC$W_LENGTH] =
```

NML\$LOGOPS
V04-000

NML Logging data base operations module
NML\$REMEVT Remove event block from source buff

C 13
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 v4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 63
(24)

: 1973 1950 2 .SRCPTR [SRC\$W_LENGTH] - EVT\$K_LENGTH;
: 1974 1951 2
: 1975 1952 2 SRCPTR [SRC\$W_MSKCOUNT] =
: 1976 1953 2 .SRCPTR [SRC\$W_MSKCOUNT] - 1;
: 1977 1954 2
: 1978 1955 1 END:
: End of NML\$REMEVT

51 08 AC 56 04 007C 00000 50 14 C1 00002 50 66 D0 00007 50 56 3C 0000B 50 C0 0000E 08 BC 61 51 C2 00011 66 50 28 00014 16 14 A2 00019 16 A6 B7 0001C 04 04 0001F	.ENTRY NML\$REMEVT, Save R2,R3,R4,R5,R6 ADDL3 #20, EVT PTR, PTR MOVL SRCPTR, R6 MOVZWL (R6), BUFEND ADDL2 R6, BUFEND SUBL2 PTR, R0 MOVC3 R0, (PTR), @EVT PTR SUBW2 #20, (R6) DECW 22(R6) RET	: 1894 : 1937 : 1938 : 1942 : 1944 : 1950 : 1953 : 1955
--	---	--

: Routine Size: 32 bytes, Routine Base: \$CODE\$ + 07B0

NML
V04

NML\$LOGOPS
VO4-000

NML Logging data base operations module
NML\$REMEVT Remove event block from source buff

D 13
16-Sep-1984 00:19:25
14-Sep-1984 12:50:11

VAX-11 Bliss-32 V4.0-742
[NML.SRC]NMLLOGOPS.B32;1

Page 64
(25)

: 1980 1956 1 END
: 1981 1957 1
: 1982 1958 0 ELUDOM

! End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	1044	NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$SPLITS	40	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	2000	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[NML.OBJ]NMLLIB.L32;1	341	40	11	27	00:00.1
-\$255\$DUA28:[SHRLIB]NMALIBRY.L32;1	887	5	0	47	00:00.2
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	2	0	581	00:02.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:NMLLOGOPS/OBJ=OBJ\$:NMLLOGOPS MSRC\$:NMLLOGOPS/UPDATE=(ENH\$:NMLLOGOPS)

Size: 2000 code + 1084 data bytes

Run Time: 00:40.1

Elapsed Time: 01:39.0

Lines/CPU Min: 2931

Lexemes/CPU-Min: 12503

Memory Used: 134 pages

Compilation Complete

0284 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

